UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
INGENIERÍA TÉCNICA INFORMÁTICA DE GESTIÓN

Definición de un modelo de intercambio de datos entre SIGEs

Paloma Galeote Bajo

David Díez Cebollero

27/05/2009

Agradecimientos

Durante todo el tiempo en el que he estado realizando este Proyecto Fin de Carrera y durante los años de Universidad siempre he tenido personas a mí alrededor que me han apoyado y animado para seguir adelante. Ahora me gustaría agradecerles sus esfuerzos y consejos, porque sin su ayuda todo habría resultado mucho más difícil.

En primer lugar quiero mencionar a mi familia, que ha estado preocupándose desde el primer día y me ha brindado su apoyo incondicional, a mis padres que desde que me matriculé han confiado en que podría hacerlo, a mis hermanas y cuñados por armarse de paciencia en mis malos momentos y por hacerme disfrutar de los buenos.

Se lo quiero agradecer especialmente a Manu, por estar ahí, por estudiar conmigo y por ayudarme. Muchas gracias.

A toda la gente importante que me ha apoyado en la carrera, con los que he compartido más de un millón de momentos buenos y los que quedan, a María, Manu H., Manu de la T., Alberto, Jesús, Tati y Sandra.

A los que se me han ayudado con sus conocimientos, Jesús y Fer, por las miles de cosas que me han explicado y sobre todo por su tiempo.

También me gustaría agradecérselo a todos mis tutores, Laura, Dani y David, por su ayuda durante todo el proyecto, en especial a David por su trabajo constante y su paciencia.

Índice de contenidos

Índice de figuras

Índice de tablas

Glosario de términos

ARCE	Aplicación en Red para Casos de Emergencias.
API	Application Programming Interface.
CCAA	Comunidades Autónomas.
DGPCE	Dirección General de Protección Civil y Emergencias.
DOM	Document Object Model.
DTD	Document Type Definition.
FTP	File Transfer Protocol.
HTML	Hipertext Markup Language.
IDE	Integrated Development Environment.
JAXB	Java Architecture for XML Binding.
JAXM	Java Architecture for XML Messaging.
JAXP	Java API for XML Processing.
JAXR	Java Architecture for XML Registries.
JAX-RPC	Java Architecture for XML Reference Implementation Project.
JDBC	Java Data Base Connectivity.
JDK	Java Development Kit.
JRE	Java Runtime Environment.
JVM	Java Virtual Machine.
OMG	Object Management Group
RBS	Risk Breakdown Structure.
RF	Requisitos Funcionales.
RNF	Requisitos No Funcionales.

SAX	Simple API for XML.
SIGAME	Sistema de Gestión de Ayudas con Medios en Emergencias.
SIGE	Sistemas de Gestión de Emergencias.
SGML	Standard Generalized Language.
UML	Unified Modelling Language.
W3C	Consorcio para la Word Wile Web.
XML	Extensible Markup Language.
XSL	Extensible Style Language.
RUP	Rational Unifiel Process.

1 Introducción

La gestión de emergencias se caracteriza por la necesidad de coordinación entre distintos individuos o sistemas para resolver una amenaza, riesgo o situación de alerta; por lo tanto, el intercambio de información y la coordinación es imprescindible para resolver las emergencias de forma rápida y eficaz.

El presente proyecto pretende proporcionar mecanismos de interoperabilidad que permita la gestión conjunta de situaciones de emergencia; concretamente, nos centraremos en dos Sistemas Web de Gestión de Emergencias: ARCE (Aplicación en Red para Casos de Emergencia) y SIGAME (SIstema de Gestión de Ayudas con Medios en Emergencias). El proyecto tratará de garantizar la transformación de datos de forma adecuada, de tal manera que la información sobre la situación de emergencia existente en uno de los sistemas se vea reflejada en el otro. Este proceso debe ser bidireccional; es decir, la información presente en los distintos sistemas debe ser la misma después de la ejecución de una acción con repercusión en ambos sistemas.

En este capítulo se introducirá al lector en el entorno de desarrollo del proyecto y se expondrán los factores que han propiciado su realización. El objetivo es exponer una visión general del trabajo realizado.

1.1 Planteamiento del problema

La respuesta óptima a una situación de emergencia requiere de la colaboración entre distintos organismos y entidades. Esta colaboración implica el intercambio de información entre los participantes en el proceso de gestión. Una forma de optimizar el proceso de gestión de la emergencia se encuentra en la utilización de aplicaciones informáticas. Mediante los sistemas informáticos de gestión de emergencias se pretende informar a terceros sobre el suceso que ha ocurrido en algún lugar, de manera que estos terceros puedan ofrecer una respuesta adecuada. El problema existente es la ausencia de estándares y mecanismos universalmente aceptados para el intercambio de información. En particular, en el contexto considerado, ARCE y SIGAME son sistemas de gestión de emergencias independientes pero susceptibles de trabajar en común, motivo por el cual es

necesario proporcionar mecanismos de interoperabilidad entre los mismos.

1.2 Objetivos

El objetivo de este proyecto es proporcionar mecanismos que permitan alcanzar la interoperabilidad sintáctica entre los sistemas ARCE y SIGAME. La consecución de este objetivo principal requiere la satisfacción de una serie de objetivos secundarios:

1. Revisar tecnologías disponibles
2. Revisar el funcionamiento de los sistemas implicados.
3. Definir un modelo de información común a ambos sistemas.
4. Desarrollar mecanismos de conversión que permitan transformar la información de los sistemas al modelo común definido.
5. Definir y desarrollar validadores que permitan controlar el formato de la información intercambiada.

La consecución de todos estos objetivos específicos permitirá la consecución del objetivo principal del trabajo.

1.3 Estructura del trabajo

La presente memoria se divide en ocho capítulos. En primer lugar, y descontando el presente capítulo de introducción, se realiza el estudio de las tecnologías empleadas para la realización del proyecto y la revisión del contexto del problema. A continuación, en el capítulo tres, gestión del proyecto software, se expone el plan de trabajo y la gestión de recursos del proyecto. En el siguiente capítulo, capítulo de solución, se realiza la revisión de los sistemas implicados y la exposición de la solución elaborada. La revisión servirá para conocer las limitaciones de los sistemas existentes y determinar la forma en la cual pueden resolverse dichos problemas. Una vez definida la solución, se expone el proceso de desarrollo seguido para elaborar dicha solución. En el capítulo cinco, el capítulo de evaluación, podemos ver los casos en los que se han desarrollado las pruebas y un análisis de resaltados de estas pruebas. A continuación, veremos un capitulo de conclusiones donde se resumen las aportaciones del proyecto. El trabajo se complementa con una serie de anexos que recogen: Anexo A. Control de versiones, donde se almacenan todas las

versiones de esta memoria, la fecha y una descripción de los cambios producidos. Anexo B. Modelo de Información, en este anexo se muestran los datos que se han insertado en los sistemas y después de ejecutar el programa se han almacenado en el documento XML definido. Anexo C. Modelo de datos de partida, en este apartado se describirá las tablas de las bases de datos de ambos sistemas que utilizamos a lo largo de este proyecto.

2 Estudio del problema

Una emergencia es una situación que plantea una amenaza inmediata a la vida humana o un daño al medioambiente. Para afrontar y gestionar situaciones de la emergencia, los pasos más significativos son la coordinación y la comunicación entre el segmento afectado y aquel que intenta solventar el problema. Para ello, son necesarios unos sistemas de emergencia que puedan compartir información de manera eficaz y fiable. Por tanto, la comunicación entre los implicados en la respuesta a la emergencia resulta esencial.

Los Sistemas de Gestión de Emergencias (SIGE) son diseñados para conseguir aquella coordinación y solucionar los problemas de comunicación. Su objetivo es proporcionar canales de comunicación seguros y confiables, recopilar la información sobre la situación por la cual se inicia una incidencia y las necesidades que se precisan, manejar recursos, informar sobre las peticiones y gestionar respuestas de un modo coordinado, la planificación de entrega, las revisiones, información geográfica, etc. Si en la respuesta a la emergencia se ven involucrados distintos organismos o es necesario emplear recursos de diferentes procedencias, es imprescindible disponer de una comunicación e intercambio de información entre distintos sistemas. Esta relación es lo que entenderemos como interoperabilidad. La interoperabilidad desde un punto de vista informático, se define como la habilidad que tiene un sistema o producto para trabajar con otros sistemas o productos sin un esfuerzo especial por parte del cliente (usuario). Este concepto tiene una importancia creciente a tenor de las colecciones digitales distribuidas que utilizan distintos esquemas de metadatos. A pesar de la complejidad de este concepto y de sus múltiples implicaciones para los sistemas de recuperación de información basados en metadatos, es un concepto clave al hablar de esquemas de metadatos y de la necesidad de compatibilizar todos ellos, para una recuperación de información integral en distintas colecciones de datos y metadatos distribuidos. La interoperabilidad entre distintos esquemas de metadatos puede realizarse de diversas formas, por ejemplo a través del funcionamiento de un protocolo o bien a través del mapeo o establecimiento de correspondencias entre informaciones en diferentes formatos para la conversión de elementos de metainformación que permita hacerlos compatibles.

Para ello contamos con dos sistemas de gestión de emergencia, ARCE y SIGAME. Se trata de sistemas independientes entre sí pero con objetivos similares, lo que invita al uso combinado de los mismos. Este hecho requiere de mecanismos de intercambio de información. A continuación, y con el fin de concretar el problema, se expondrá en detalle de cada uno de los sistemas y la forma en la cual podrían trabajar en común.

2.1 El contexto del problema

Los SIGE son diseñados para conseguir una coordinación y solucionar problemas de comunicación. Sus tareas principales son proporcionar canales de comunicación seguros y confiables, recopilar la información sobre la situación por la cual se inicia una incidencia y las necesidades que se precisan, manejar recursos, informar sobre las peticiones y gestionar las respuestas de un modo coordinado, la planificación de entrega, las revisiones, información geográfica,… En el proyecto manejamos dos de estos SIGE:

1. SIGAME es una herramienta informática de comunicación que permite una mayor eficacia en la respuesta ante las emergencias, y una rápida puesta a disposición de la ayuda que se requiera por parte de las Comunidades Autónomas.
2. ARCE es un programa de cooperación Iberoamericano, para la gestión de emergencias. Es un sistema cuyo objetivo es gestionar emergencias a nivel supranacional.

A continuación, para mayor comprensión del contexto del problema se describirá en detalle cada uno de estos sistemas.

SIGAME (SIstema de Gestión de Ayudas con Medios en Emergencias)

"SIGAME es una plataforma tecnológica cuyo objetivo es facilitar la aportación de recursos ubicados fuera de una Comunidad Autónoma para hacer frente a una situación de emergencia o suceso, producida en ésta, a requerimiento del órgano competente para la gestión de la misma. Esta plataforma favorece una comunicación inter-territorial y multi-direccional para hacer más eficaz y eficiente la respuesta ante situaciones de emergencia y no interfiere con los protocolos de gestión de sucesos existentes en las diferentes Comunidades Autónomas."

SIGAME facilita la comunicación entre las comunidades autónomas para afrontar una situación de emergencia, de la forma más eficiente y eficaz al responder a las solicitudes. SIGAME da soporte a los siguientes escenarios:

- Coordinación entre comunidades con acuerdos bilaterales. Algunas comunidades establecen acuerdos como: proximidad geográfica,… en estos casos es necesario establecer un canal directo de comunicación entre ellos.
- Coordinación de la ayuda supra-comunitaria supervisados por la DGPCE (Dirección General de Protección civil y Emergencias). Cuando una comunidad pide ayuda a otras comunidades autónomas sin mediar acuerdos bilaterales, es necesario coordinar y garantizar que la ayuda cubre las necesidades de la comunidad solicitante. Esto es responsabilidad de la DGPCE, es decir, debe de hacerse cargo de que las comunidades oferten los recursos disponibles.
- Supervisión de recursos en la cooperación entre comunidades. Es necesario desarrollar mecanismos de modo que las comunidades en todo momento puedan conocer el estado de los recursos que vienen enviados por otras comunidades.

Como características específicas del sistema SIGAME se debe mencionar que dicho sistema está implementado en Java emplea MySQL como gestor de base de datos. Para una mayor información, podemos consultar www.sigame.es.

ARCE (Aplicación en Red)

ARCE es un sistema Web diseñado para mejorar la respuesta conjunta ante situaciones de emergencia entre todos los miembros de la Asociación Iberoamericana de Organismos Gubernamentales de Defensa y Protección Civil que involucra a 21 países. El sistema proporciona mecanismos para notificar una emergencia, pedir recursos y ofrecer asistencia. Cuando una emergencia ocurre, el país afectado, utiliza ARCE para informar a la asociación sobre la situación. Cuando se requiere asistencia, los recursos son clasificados según un glosario multilingüe incluido en el sistema, la petición de ayuda es dada de alta,

los asociados son notificados por correo electrónico para que puedan acceder al sistema y ver qué recursos son necesarios y cómo pueden ayudar. Para cada emergencia, la aplicación proporciona información actualizada sobre qué recursos son necesarios, las cantidades iniciales y cuántos elementos han sido ya facilitados, permitiendo así que cada asociado decida cómo contribuir.

ARCE se puede usar tanto en situación de emergencia como en situación normal, en un momento en el que no se considere emergencia, se podrá publicar y visualizar noticias relevantes para los organismos asociados, e intercambiar mensajes de información entre usuarios autorizados en ARCE. Los organismos asociados que quieran responder a la solicitud de ayuda exterior, tanto la primera solicitud como la detallada, realizada por un organismo asociado, podrán ofrecer aportaciones sobre los medios solicitados visualizando en todo momento las cantidades solicitadas por el organismo asociado demandante de la ayuda. Este ofrecimiento podrá ser aceptado, denegado o modificado por el organismo asociado que ha solicitado la ayuda exterior. En el caso de que se modifique el ofrecimiento el país aportante deberá dar el visto bueno de la modificación antes de considerar el ofrecimiento como aceptado. Como características específicas del sistema de emergencias ARCE se debe mencionar que dicho sistema está implementado en Java sobre servidores Zope [1] y emplea como base de datos PostgreSQL.

2.2 El estado de la cuestión

Para llevar a cabo la implementación de este proyecto se han empleado varias tecnologías y conceptos. En cada apartado se irá mostrando al lector una descripción de la tecnología, características principales, justificación de cómo y por qué se han utilizado en el presente proyecto, y unas referencias bibliográficas para que se pueda ampliar la información aquí recogida.

El primer apartado tiene como objetivo presentar al lector un estudio de las bases de datos [página18] relacionales así como concretamente las bases de datos que se utilizarán en el proyecto, Mysql (sistema gestor en el cual se desarrolla SIGAME), y PostgreSQL (el cual es la base de ARCE). A continuación, veremos en profundidad una descripción del

lenguaje de marcado o XML [página 21] empleado para la comunicación de ambas bases de datos. El siguiente apartado se centra en la tecnología Java [página 25] empleada para la implementación del proyecto. Los dos siguientes apartados describen el modo de comunicar lo anteriormente visto, en primer lugar veremos la relación entre Java y las bases de datos [página 28] y a continuación, en el punto "Java y XML" el modo de comunicar Java y el lenguaje de marcado XML [página 30]. Por último, se realiza un estudio del lenguaje UML [página 36] (Lenguaje de Modelado Unificado), puesto que a lo largo de la presente memoria se mostrarán diagramas que se explicarán teóricamente en este apartado.

Bases de Datos

Una base de datos [2] se define como:

"Colección o depósito de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción, únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación comunes y bien determinados recuperación, determinados, habrán de ser capaces de conservar la integridad, disponibilidad y confidencialidad del conjunto de los datos".

A continuación, se realiza una revisión sobre el concepto de bases de datos relaciones y los gestores de bases de datos implicados en nuestro proyecto.

Bases de Datos Relacionales

El modelo de Bases de Datos Relacionales [3] es el más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se

conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por *registros* (las filas de una tabla), que representarían las tuplas, y *campos* (las columnas de una tabla). En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

En el presente proyecto se han utilizado dos bases de datos existentes para los dos sistemas de emergencia, en el caso del sistema SIGAME se manipula MySql, y en el caso de ARCE es PostgreSQL, vamos a ver una breve introducción a estas bases de datos.

1. MySql

MySQL [4] es un sistema de bases de datos que se puede encuadrar dentro de la categoría open-source (disponible y abierto a modificaciones). Aunque open-source no siempre implica que sea gratuito MySql si lo es.

El origen de MySQL se remonta a la década de los ochenta. Michael Widenius, también conocido como *Monty*, un joven programador que realizaba complejas aplicaciones en lenguaje BASIC, al no encontrar un sistema de almacenamiento de archivos que le resultara satisfactorio, pensó en construir el suyo propio. Años después, en 1995, y en colaboración con David Axmark, Widenius desarrolló un producto que básicamente era el resultado de sus investigaciones, más dos aportaciones nuevas: el uso del lenguaje SQL y la accesibilidad a través de Internet. Así nació MySQL y también la empresa MySQL AB. Aparte de las características que definen MySQL como programa *open-source*, existen aspectos que lo diferencian de otros productos como, por citar uno conocido, Access. Los atributos a los que hacemos referencia son, en líneas muy generales:

1. Posibilidad de crear y configurar usuarios, asignando a cada uno de ellos permisos diferentes.
2. Facilidad de exportación e importación de datos, incluso de la base de datos completa.

3. Posibilidad de ejecutar conjuntos de instrucciones guardadas en ficheros externos a la base de datos.

No nos centraremos en las características concretas de MySql, ya que no es objeto del proyecto. Únicamente destacaremos la característica de MySQL de ser una base de datos muy rápida en la lectura aunque puede provocar problemas de integridad en entornos de alta concurrencia. En aplicaciones Web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

2. PostgreSQL

PostgreSQL [5] ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos funciones y otras características aportan potencia y flexibilidad adicional (restricciones, disparadores, reglas, e integridad transaccional).

Estas características colocan a PostgreSQL en la categoría de las Bases de Datos identificadas como *objeto-relacionales*.

PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl y python). Durante el desarrollo de PostgreSQL95 se hizo hincapié en identificar y entender los problemas en el código del motor de datos. Con PostgreSQL, el énfasis ha pasado a aumentar características y capacidades, aunque el trabajo continúa en todas las áreas. Las principales mejoras en PostgreSQL incluyen:

- Los bloqueos de tabla han sido sustituidos por el control de concurrencia multi-versión, el cual permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros, y permite copias de seguridad en caliente desde pg_dump mientras la base de datos

permanece disponible para consultas.

- Se han implementado importantes características del motor de datos, incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (constraints) y disparadores (triggers).
- Se han añadido funcionalidades en línea con el estándar SQL92, incluyendo claves primarias, identificadores entrecomillados, forzado de tipos cadenas literales, conversión de tipos y entrada de enteros binarios y hexadecimales.
- Los tipos internos han sido mejorados, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.
- La velocidad del código del motor de datos ha sido incrementada aproximadamente en un 20-40%, y su tiempo de arranque ha bajado el 80% desde que la versión 6.0 fue lanzada.

3. Lenguaje de Marcado XML (Extensible Markup Language)

XML, o Lenguaje de Marcado Extensible [6] (Extensible Markup Language), es un lenguaje de marcado que puede usar para crear sus propias etiquetas. Fue creado por el Consorcio para la World Wide Web (W3C) para superar las limitaciones de HTML (HiperText Markup Language), el Lenguaje de Marcado de HiperTexto que es la base para todas las páginas Web.

La primera definición de XML fue la de "Sistema para definir, validar y compartir formatos de documentos en la Web". Para crear XML se tomaron las mejores partes tanto del lenguaje SGML (antecesor tanto de XML como de HTML, Standard Generalizad Markup Language) como del HTML. La diferencia fundamental entre HTML y XML es que el primero estaba orientado a la presentación de datos, mientras que XML está orientado a los datos en sí mismos, por lo que cualquier software informático trabajará mejor con XML. Sin duda, esta diferencia es fundamental para los nuevos desarrollos de la Web donde se da suma importancia al contenido de los datos y su tratamiento, y no sólo a su presentación.

HTML era, principalmente, un lenguaje de presentación que definía un conjunto de etiquetas y atributos válidos y que ofrecía un significado visual para cada elemento del lenguaje, por el contrario, XML no define las etiquetas ni cómo se utilizan, sino que ofrece

un escaso número de reglas sintácticas para poder crear documentos. Así pues, XML no es un lenguaje, sino un metalenguaje o lenguaje para definir otros lenguajes. XML no sustituye a HTML puesto que sirven para cosas distintas: una cosa es presentar la información (para lo que sigue siendo válido HTML) y otra bien distinta es representar e intercambiar los datos de forma independiente a su presentación (que es para lo que sirve XML). Además, XML no sólo se aplica en Internet, sino que se propone como un lenguaje de bajo nivel para intercambio de información estructurada entre distintas plataformas. Se puede utilizar en bases de datos, hojas de cálculo, editores de texto, etc. y no sólo en la Web. HTML y XML son, pues, dos lenguajes complementarios.

1. Objetivos del XML

Según la especificación, los objetivos de diseñar XML fueron los siguientes:

- XML debe ser directamente utilizable en Internet
- XML debe soportar una amplia variedad de aplicaciones
- XML debe ser compatible con SGML
- Debería se sencillo escribir programas que procesaran documentos XML
- El número de las características opcionales en XML debería ser el mínimo posible, a ser posible cero
- Los documentos XML deberían ser legibles por las personas y razonablemente claros
- El diseño de XML debe ser rápido
- XML debería ser simple, pero perfectamente normalizado
- Los documentos XML deben ser de fácil creación
- La concisión de las marcas XML tiene una importancia mínima

XML es un lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo. La información estructurada presenta varios contenidos (texto, imágenes, audio, etc.) y formas: hojas de cálculo, tablas de datos, libretas de direcciones, parámetros de configuración, dibujos técnicos, etc. La forma da alguna indicación de qué papel puede

jugar el contenido (por ejemplo, el contenido de una sección encabezada con un significado difiere del contenido de una nota a pie de página, lo que significa algo diferente que el contenido de un pie de foto o el contenido de una tabla de datos). Más o menos todos los documentos tienen la misma estructura.

2. Principales diferencias entre XML y HTML:

Es importante comprender por qué se ha creado XML. XML se ha creado para enriquecer la estructura de los documentos que pueden ser usados en la Web, puesto que las otras alternativas viables, HTML y SGML, no eran demasiado prácticas para este propósito.

Las diferencias fundamentales de XML con respecto a HTML son las siguientes:

- No requiere DTD (Document Type Definition),
- El XML tiene punteros a la estructura de los datos, lo que ahorra tiempo y simplifica el software de aplicación.
- XML no dispone de soporte para excepciones, por lo que cada etiqueta realiza siempre la misma función.
- Posee independencia de los navegadores y del sistema de objetos, porque en lugar de añadir etiquetas de presentación al documento se remite a una hoja de estilo realizada en XSL (Extensible Style Languaje).

3. DTDs y XML Schema

La necesidad de jerarquizar y estructurar correctamente la información, no sólo para almacenarla, sino también para acceder a ella, se ha convertido en una labor que ha cobrado especial relevancia en los últimos años, en los que se han producido importantes avances en este campo. Existen las denominadas DTDs (Definición de Tipo de Documento) y los XML Schema, en este apartado se hará una breve introducción de ambas, para poder decidir cual utilizar en el proyecto.

En los siguientes párrafos se realiza un breve estudio sobre los DTDs y los XML Schema.

Los DTDs cumplen las siguientes funciones:

- Especifica la clase de documento.
- Describe un formato de datos.
- Usa un formato común de datos entre aplicaciones.
- Verifica los datos al intercambiarlos.
- Verifica un mismo conjunto de datos.

Así pues, una DTD especifica la clase de documento XML. Una DTD sólo indica qué elementos o qué atributos,… tiene un documento y como se anidan, pero no dice nada acerca de tipo de dato. El único tipo de datos que se conoce es CDATA (texto plano), por tanto, las DTDs se quedan algo cortas y nosotros en el proyecto necesitamos algo más potente, ya que queremos almacenar datos de las bases de datos.

Por el motivo expuesto en el párrafo anterior se opta por el uso de XML Schema, al igual que las DTDs, los Xml Schemas describen el contenido y la estructura de la información, pero de una forma más precisa. Los esquemas indican tipos de datos, número mínimo y máximo de ocurrencias y otras características más específicas.

Según la Especificación del W3C Xml Schema (http://www.w3.org/XML/Schema), los esquemas expresan vocabularios compartidos que permiten a las máquinas extraer las reglas hechas por las personas. Los esquemas proveen un significado para definir la estructura, contenido y semántica de los documentos XML.

Un esquema XML es algo similar a un DTD, es decir, define que elementos puede contener un documento XML, cómo están organizados y qué atributos y qué tipo pueden tener sus elementos, pero la utilización de esquemas ofrece nuevas posibilidades en el tratamiento de los documentos.

La ventaja de usar Schemas con respecto a los DTDs son:

- Usan sintaxis de XML, al contrario que los DTDs.
- Permiten especificar tipos de datos.
- Son extensibles, es decir permiten crear nuevos elementos.

Puede utilizar Espacios de Nombre ya definidos, los espacios de nombre son (a grandes rasgos) un medio para organizar clases dentro de un entorno, agrupándolas de un modo

más lógico y jerárquico. Contiene:

- *ElementType:* Define el tipo y contenido de un elemento, incluyendo los subelementos que pueda contener.
- *AttributeType*: Asigna un tipo y condiciones a un atributo.
- *Attribute:* declara que un atributo previamente definido por AttributeType puede aparecer como un atributo de un elemento determinado.
- *Element*: declara que un element previamente definido por ElementType puede aparecer como contenido de otro elemento determinado.

Para desarrollar el proyecto se ve más conveniente formar la estructura de la base de datos con Xml Schema, ya que además de las características anteriores tiene más tipos de datos y más flexibilidad.

4. Tecnología JAVA

Java *[7]* surgió en 1991 cuando un grupo de ingenieros de *Sun Microsystems* trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.

Debido a la existencia de distintos tipos de CPUs y a los continuos cambios, era importante conseguir una herramienta independiente del tipo de CPU utilizada. Desarrollaron un código "neutro" que no dependía del tipo de electrodoméstico, el cual se ejecutaba sobre una "*máquina hipotética o virtual*" denominada ***Java Virtual Machine*** (***JVM***). Era la *JVM* quien interpretaba el código neutro convirtiéndolo a código particular de la CPU utilizada. A pesar de los esfuerzos realizados por sus creadores, ninguna empresa de electrodomésticos se interesó por el nuevo lenguaje.

Como lenguaje de programación para computadores, *Java* se introdujo a finales de 1995. La clave fue la incorporación de un intérprete *Java* en la versión 2.0 del programa Netscape Navigator, produciendo una verdadera revolución en Internet. *Java 1.1 ap*areció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje. *Java 1.2*, más tarde rebautizado como *Java 2*, nació a finales de 1998.

Al programar en *Java* no se parte de cero. Cualquier aplicación que se desarrolle "cuelga" en un gran número de *clases* preexistentes. Algunas de ellas las ha podido hacer el propio usuario, otras pueden ser comerciales, pero siempre hay un número muy importante de clases que forman parte del propio lenguaje (el ***API*** o ***Application Programming Interface*** de ***Java***).

El principal objetivo del lenguaje *Java* es llegar a ser el "nexo universal" que conecte a los usuarios con la información, esté ésta situada en el ordenador local, en un servidor de *Web*, en una base de datos o en cualquier otro lugar.

La compañía *Sun* describe el lenguaje *Java* como "*simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico*". Además de una serie de halagos por parte de *Sun* hacia su propia criatura, el hecho es que todo ello describe bastante bien el lenguaje *Java*, aunque en algunas de esas características el lenguaje sea todavía bastante mejorable.

1. El entorno de desarrollo JAVA

Existen distintos programas comerciales que permiten desarrollar código *Java*. La compañía *Sun*, creadora de *Java*, distribuye gratuitamente el *Java(tm) Development Kit* (***JDK***). Se trata de un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en *Java*. Incorpora además la posibilidad de ejecutar parcialmente el programa, deteniendo la ejecución en el punto deseado y estudiando en cada momento el valor de cada una de las variables (con el denominado *Debugger*). Cualquier programador con un mínimo de experiencia sabe que una parte muy importante (muchas veces la mayor parte) del tiempo destinado a la elaboración de un programa se destina a la *detección y corrección de errores*. Existe también una versión reducida del *JDK*, denominada ***JRE*** (*Java Runtime Environment*) destinada únicamente a ejecutar código *Java* (no permite compilar).

Los ***IDE***s (*Integrated Development Environment*), tal y como su nombre indica, son entornos de desarrollo integrados. En un mismo programa es posible escribir el código *Java*, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Algunos incluyen una herramienta para realizar *Debug* gráficamente, frente a la versión que incorpora el *JDK* basada en la utilización de una consola (denominada habitualmente ventana de comandos

de MS-DOS, en ***Windows NT/95/98***). Estos entornos integrados permiten desarrollar las aplicaciones de forma mucho más rápida, incorporando en muchos casos librerías con *componentes* ya desarrollados, los cuales se incorporan al proyecto o programa. Como inconvenientes se pueden señalar algunos fallos de compatibilidad entre plataformas, y ficheros resultantes de mayor tamaño que los basados en clases estándar.

En la Ilustración 1 Java Platform, Standard Edition (Java SE) se muestra de manera esquemática la arquitectura de la plataforma Java, la cual contiene los elementos de los que hemos hablado anteriormente, y cuya característica más potente es la posibilidad de ejecutar programas creados en ella, independiente del tipo de arquitectura o dispositivos computacionales utilizados.

Java está constituida por un conjunto de especificaciones que definen todos y cada una de las partes de la plataforma, y una serie de implementaciones de estas especificaciones.

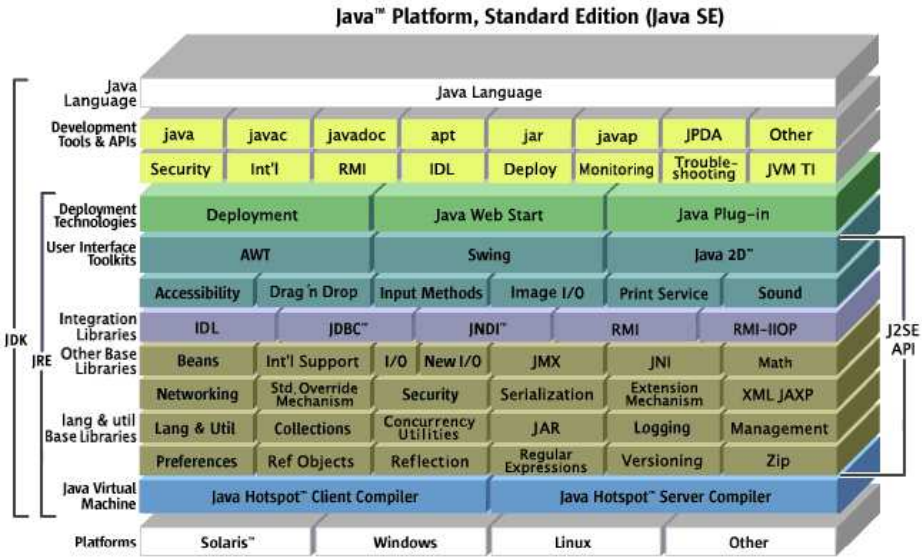


Ilustración 1 Java Platform, Standard Edition (Java SE)

Como vemos en la figura, los grandes bloques en los que está constituida la plataforma son varios programas, cada uno de los cuales proporciona una funcionalidad distinta de sus capacidades totales. En el presente proyecto utilizamos por ejemplo JDBC que es una

librería para la iteración con las bases de datos, también posee Javac que se trata de un compilador que convierte el código original Java en Java bytecode,…

A medida que vayamos utilizando estos componentes los iremos nombrando.

La figura está presente en la siguiente página, junto con toda la información de la misma, http://java.sun.com/javase/6/docs/.

5. JAVA y las Bases de Datos

En esta sección se tratará de introducir la manera de conectar **Java y las Bases de Datos** [8]**,** es decir, se hablará de la **API JDBC** (Java Database Connectivity), se intentará estudiar el tema con cierta profundidad puesto que es una parte fundamental del proyecto**.**

JDBC es un API de Java para acceder a sistemas de bases de datos, y prácticamente a cualquier tipo de dato tabular. El API JDBC consiste de un conjunto de clases e interfaces que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea.

La arquitectura de JDBC está basada en un conjunto de interfaces y clases de java que permiten conectarse con un motor de la base de datos no importando de que tipo sea, crear y ejecutar sentencias SQL, recuperar y modificar datos de una base de datos, esta tecnología se detalla con profundidad en la Web: http://java.sun.com/jdbc/.

Las características más importantes de ésta API son:

- Posibilidad de utilizar lenguaje SQL.
- Acceso sencillo a todas las funciones de SQL.
- Los resultados devueltos por SQL pueden ser tratados como objetos, de tal manera que si hay errores, se pueden tratar como excepciones.

En la siguiente figura, Ilustración 2 Funcionamiento del driver JDBC, se ven las operaciones que pueden realizarse en una base de datos. Cada caja representa una clase o interfaz de JDBC que tiene un rol en el acceso a la base de datos relacional.

El trabajo comienza con la clase DriverManager, que la que establece las conexiones

con las fuentes de datos, mediante los driver o controladores JDBC.

Los driver de bases de datos JDBC se definen mediante clases que implementan la interfaz Driver. Cada driver sabe cómo convertir peticiones SQL para cada base de datos concreta. Si no disponemos del driver adecuado, no podemos conectarnos a la base de datos, y el JDBC depende de las implementaciones concretas de cada fabricante. Por tanto, lo primero que se ha de hacer es cargar el driver adecuado.

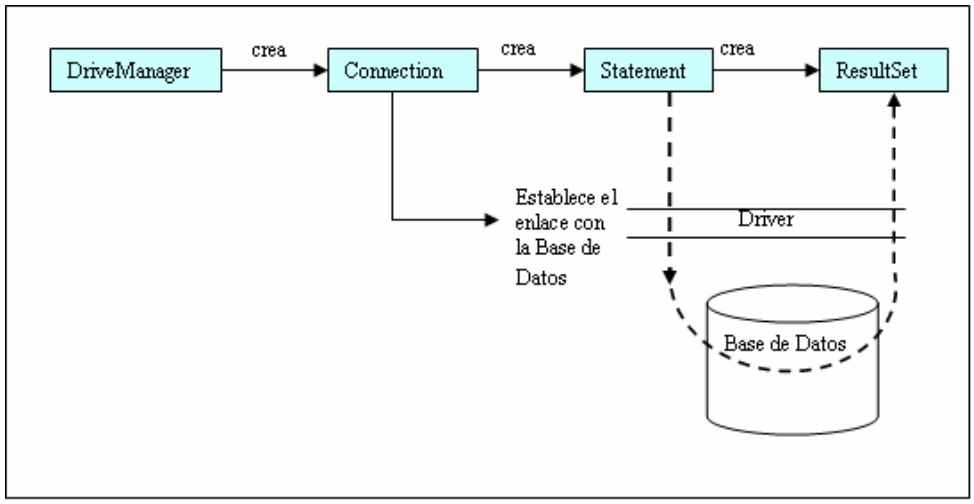


Ilustración 2 Funcionamiento del driver JDBC

Podemos describir un funcionamiento, basado en los siguientes pasos:

En primer lugar, cargaremos el driver JDBC para conseguir una conexión con las bases de datos y así poder acceder a los datos, crearemos una instancia por base de datos. A continuación, debemos construir una URL para cada una en la que se define la forma de conexión a la base de datos, el servidor en el que está, el nombre de la misma y un usuario. Estos dos primeros pasos reflejan lo que sería la primera clase (DriveManager) que aparece en la Ilustración 2. El siguiente paso es crear la instancia un objeto de tipo Connection, será nuestra conexión con la base de datos, se representa en la ilustración por la segunda caja. Una vez creada la conexión, podemos crear sentencias o ejecutar consultas desde nuestro programa java para interactuar con la base de datos, esto lo realizaremos gracias al objeto

Statement.

Por último, para ejecutar una consulta con el objeto Statement empleamos el método executeQuery(), las consultas de recuperación de tuplas o registros se ejecutan con este método. El método devuelve un objeto *ResultSet,* el cual puede usarse para acceder a cada uno de los registros devueltos. Al acabar la interacción con la base de datos debemos liberar los objetos ResultSet, Statement, y Connection.

6. JAVA y XML

Para tratar las conexiones entre **Java y XML** [9] tenemos las APIs que nos permiten escribir aplicaciones Web completamente en el lenguaje Java. Se dividen en dos categorías:

- Orientadas al documento: tratan directamente con los documentos XML.
  - API Java para Procesar XML (JAXP) — procesa documentos XML usando varios analizadores.
  - Arquitectura Java para Uniones XML (JAXB) — mapea elementos XML a clases del lenguaje Java.
- Orientadas al procedimiento: tratan con procedimientos.
  - API Java para Mensajería XML (JAXM) — envía mensajes SOAP sobre Internet de una forma estándar.
  - API Java para Registros XML (JAXR) — proporciona una forma estándar para acceder a registros de negocios que comparte información.
  - API Java para RPC basado en XML (JAX-RPC) — envía llamadas a métodos SOAP a partes remotas sobre Internet y recibe los resultados.

Las características más importantes de los APIs de Java para XML son:

- Todos soportan los estándares de la industria, así que aseguran la interoperabilidad.
- Permiten gran flexibilidad.

Puesto que nuestro objetivo es generar clases java a partir de un documento Xml Schema, para poder realizar la migración, lo mejor será utilizar JAXB, a continuación

estudiamos esta tecnología.

JAXB proporciona una manera rápida y conveniente de crear uniones de dos vías entre los documentos XML y los objetos Java. Dado un esquema, que especifica la estructura de los datos XML, el compilador JAXB genera un conjunto de clases de Java que contienen todo el código para analizar los documentos XML basados en el esquema. Una aplicación que utilice las clases generadas puede construir un árbol de objetos Java que representa un documento XML, manipular el contenido del árbol y regenerar los documentos del árbol, todo ello en XML sin requerir que el desarrollador escriba código de análisis y de proceso complejo.

- Usar JAXB para una aplicación de proceso de datos tiene muchos beneficios porque una aplicación JAXB:
- Usa Tecnología Java y XML
- Garantiza Datos Válidos
- Es Rápida
- Es Fácil de Usar
- Puede Restringir Datos
- Es Personalizable
- Es Extensible
- Esta sección explica todas esas cualidades en más detalle.

1. Las aplicaciones JAXB usan Tecnología JAVA y XML

Las razones más importantes para utilizar JAXB son que las aplicaciones de JAXB están escritas en el lenguaje de programación de Java y pueden procesar datos XML.

XML es una forma estándar industrial e independiente del sistema de representar datos. Los datos que se representan usando XML se pueden publicar en múltiples medios porque describe la estructura de los datos, no su formato. Los datos de XML se pueden pasar entre aplicaciones porque la estructura de los datos se puede especificar en un esquema, lo que permite que un analizador de sintaxis valide y procese los datos que siguen el esquema. XML utiliza el esquema para definir nuestras propias etiquetas para

describir nuestros datos. Los datos XML son fáciles de trabajar porque están escritos en un formato de texto simple, legible por los seres humanos y el software de edición de texto.

Las aplicaciones escritas en el lenguaje de programación de Java son portables: Cualquier sistema con una máquina virtual Java puede ejecutar los bytecode producidos compilando una aplicación Java.

JAXB proporciona un puente entre estas dos tecnologías complementarias. JAXB incluye un compilador que asocia un esquema a un conjunto de clases Java. Una vez que tengamos nuestras clases, podremos construir las representaciones de objetos Java de los datos XML que siguen las reglas que el esquema define. JAXB permite que creemos los objetos Java en el mismo nivel conceptual que los datos XML. Una vez que tengamos nuestros datos en la forma de objetos Java, es fácil acceder a ellos. Además, después de trabajar con los datos, podemos escribir los objetos Java en un nuevo documento XML.

2. Las aplicaciones JAXB garantizan datos válidos

Es imposible utilizar JAXB para crear un árbol de objetos Java de un documento XML que sea inválido con respecto al esquema usado para crear las clases.

3. Las aplicaciones JAXB son rápidas

Dos APIs de uso general para analizar XML son SAX (API simple para XML) y DOM (modelo del objeto del documento). Un analizador de sintaxis de SAX es un analizador de sintaxis dirigido por eventos, no salva ninguna parte del documento en memoria. Un analizador de sintaxis de DOM construye una estructura de datos del documento en la memoria cuyo contenido puede ser manipulado, pero es mucho más lento que un analizador de sintaxis SAX.

JAXB hace más rápidamente el análisis porque las clases generadas están precompiladas y contienen la lógica del esquema, de tal modo que evitan la interpretación dinámica que un analizador de sintaxis SAX debe realizar.

4. Las aplicaciones JAXB son fáciles de crear y de usar

JAXB genera automáticamente el código que podemos personalizar para que realice la conversión de contenidos por nosotros.

5. Las aplicaciones JAXB pueden convertir datos

Podemos incorporar cualquier tipo de datos que deseemos entre dos etiquetas, tales como números enteros o cadenas, mientras la estructura del documento esté conforme con la especificación del DTD o el esquema.

6. Las aplicaciones JAXB pueden personalizarse

7. Las aplicaciones JAXB son flexibes

Una vez que hayamos generado las clases Java, podremos utilizarlas sin modificaciones, o subclasificarlas para proporcionar funcionalidades adicionales. Los desarrolladores de JAXB diseñaron el proceso de unión para hacer que la derivación de subclases sea sencilla.

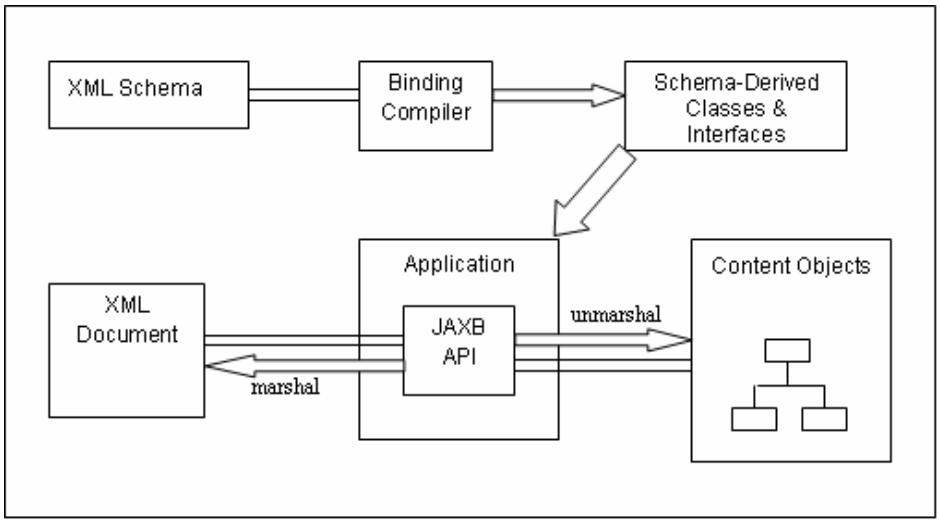


Ilustración 3 Funcionamiento de JAXB

Una vez tenemos el XML Schema ya ponemos generar los ficheros fuente de Java, tras compilar los ficheros obtenemos las clases Java con las que podremos manipular los datos que posteriormente leeremos o escribiremos en los documentos XML.

El primer paso sería construir el Xml Schema conforme a los documentos XML que vamos a recibir. Este paso se explicará mas adelante con nuestro Xml Schema del proyecto. Se podría utilizar igualmente un DTD.

Una vez tenemos el Xml Schema estamos preparados para ***Generar las Clases Java,***

para este punto debe estar configurado el classpath correctamente. Se siguen los siguientes pasos:

Ejecutamos el compilador del Xml Schema y compilamos los ficheros fuente de las clases, ya que el punto anterior nos ha generado ficheros ".class"

Con las clases que hemos generado ya podemos manejar los datos como clases java.

Construir Representaciones de datos

Las clases que genera el compilador de esquema implementan y extienden las clases e interfaces del marco de trabajo de unión. Este marco es el API de tiempo de ejecución que usan las clases generadas para soportar tres operaciones primarias:

Desempaquetar: el proceso de producir un árbol de contenidos desde un documento XML.

Validación: el proceso de verificar que la representación de objetos Java esta conforme con las reglas especificadas en el DTD o Xml Schema.

Empaquetar: el proceso de producir un documento XML desde clases Java.

Para realizar estas operaciones, cada clase generada contiene métodos para empaquetar datos, validar contenidos, y extienden métodos del marco de trabajo de unión que realizan el empaquetamiento.

1. Desempaquetar

Con los métodos unmarshall, podemos construir un árbol objetos Java desde documentos XML que son ejemplares del esquema usado para generar las clases. El árbol de objetos construido con JAXB se llama un árbol de contenido. Cada objeto del árbol corresponde a un elemento del documento XML. De forma semejantemente, cada objeto del árbol es un ejemplar de una clase del conjunto de clases generadas. También podemos construir un árbol de contenido ejemplarizando objetos de las clases porque el árbol de contenido une el documento y las clases.

2. Validación

El proceso de desempaquetamiento realiza validación mientras está construyendo el árbol de contenido, por eso es imposible desempaquetar un documento XML a un árbol de

contenido que es inválido con respecto al DTD o al XML Schema.

3. Empaquetar

Tanto si construimos el árbol de contenido usando desempaquetamiento o ejemplarización, podemos empaquetar el árbol a un nuevo documento XML usando lo método marshall. Esto significa que JAXB también permite que creemos nuevos documentos XML que son válidos con respecto al DTD / XML Schema fuente. El proceso de empaquetado comprueba si el árbol de contenido se ha validado antes de empaquetarlo en caso de que hayamos realizado cambios a los objetos del árbol. Así pues, igual que es imposible desempaquetar un documento inválido, es imposible empaquetar un árbol de contenido inválido.

La siguiente imagen nos muestra las maneras de construir representaciones de datos:

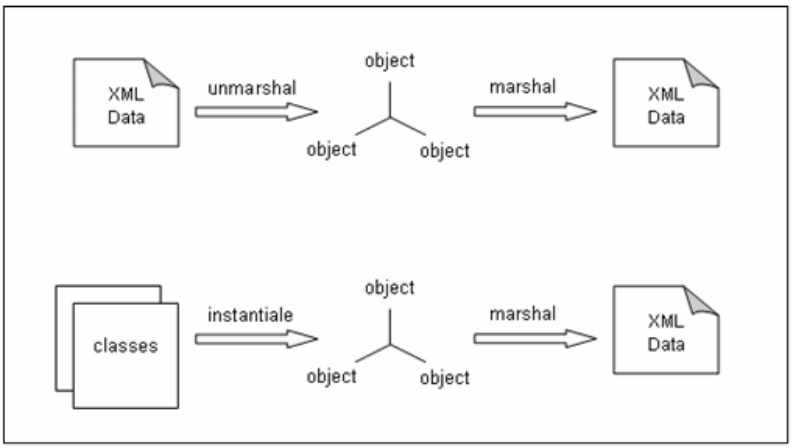


Ilustración 4 Construcción de representación de datos con JAXB

Lo primero es crear un árbol de objetos instanciando las clases con el compilador de JAXB, en la imagen se representa por la flecha “unmarshall”, seguidamente, es necesario el cast al elemento raíz de las clases generadas y luego realizaría el método un marshaller con la ruta del fichero XML, donde lo leerá, y se creará una estructura árbol. También tenemos el objeto “ObjectFactory” que es un objeto que se encarga de la creación de las instancias de las distintas clases, le creamos por tanto una instancia. Por último, si tenemos que escribir lo manipulado otra vez en un fichero XML, debemos de indicarle donde se encuentra, tal y como se hace al leer, sólo que ahora es el fichero de salida, esto es lo que

se representa como la flecha “marshall”.

7. Lenguaje Unificado de Modelado

UML [10] es un lenguaje para expresar diseños de software orientado a objetos. Sus siglas significan, en español, Lenguaje Unificado de Modelado. No es la única notación que existe, pero es el estándar actual del llamado Object Management Group (OMG). Por tanto, conviene saber cómo expresarse en este lenguaje, advirtiendo que los lenguajes son dinámicos y que, a veces, no se utilizan de la misma forma por diferentes autores.

UML se expresa a través de elementos de construcción, de relaciones y de diagramas que contienen elementos y relaciones. Conocer esta estructura general ayuda a la comprensión del lenguaje en su conjunto y facilita prescindir de los detalles, hasta que no sean necesarios.

A continuación vamos a ver sus tres bloques constructivos principales, elementos, relaciones y diagramas.

1. Elementos

Hay cuatro tipos de elementos en UML:

- Elementos estructurales.
- Elementos de comportamiento.
- Elementos de agrupación.
- Elementos de anotación.

1. Elementos estructurales

Los elementos estructurales son la parte estática de los modelos de UML. Representan cosas que son conceptuales o materiales. Hay siete tipos de elementos estructurales: clases, interfaces, colaboraciones, casos de uso, clases activas, componentes y nodos.

2. Elementos de comportamiento

Los elementos de comportamiento son las partes dinámicas de los modelos. Representan comportamiento en el tiempo y en el espacio. Hay dos tipos de elementos de comportamiento: interacciones y máquinas de estados.

3. Elementos de agrupación

Los elementos de agrupación son las partes organizativas de los modelos de UML, es decir, las cajas en las que puede descomponerse un modelo. Sólo hay un elemento de agrupación: el paquete.

4. Elementos de anotación

Los elementos de anotación son la parte explicativa de los modelos de UML. Son comentarios que se pueden aplicar para describir, clarificar y hacer observaciones sobre cualquier elemento de un modelo. El principal elemento de anotación es la nota.

2. Relaciones

Hay cuatro tipos de relaciones en UML: dependencia, generalización, asociación y realización.

1. Dependencia

Una dependencia es una relación semántica entre dos elementos, en la cual un cambio a un elemento (el elemento independiente) puede afectar a la semántica del otro elemento (el elemento dependiente).

2. Asociación

Una asociación es una relación estructural que describe un conjunto de enlaces, los cuales son conexiones entre objetos. La agregación es un tipo especial de asociación, que representa una relación estructural entre un todo y sus partes.

- **Asociaciones y agregaciones**

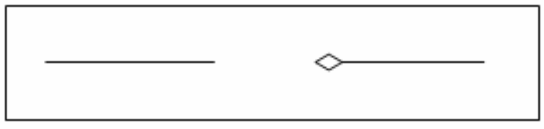


Ilustración 5 Asociaciones y agregaciones

3. Generalización

Una generalización es una relación de especialización /generalización en la cual los objetos del elemento especializado (el hijo) pueden sustituir a los objetos del elemento

general (el padre). De esta forma, el hijo comparte la estructura y el comportamiento del padre.

4. Realización

Una realización es una relación semántica entre clasificadores, en donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá. Se pueden encontrar relaciones de realización en dos sitios: entre interfaces y las clases y componentes que las realizan, y entre los casos de uso y las colaboraciones que los realizan.

3. Diagramas:

Un diagrama es la representación gráfica de un conjunto de elementos, en general visualizado como un grafo conexo de nodos (elementos) y arcos (relaciones). Los diagramas se dibujan para visualizar un sistema desde diferentes perspectivas, de forma que un diagrama es una proyección de un sistema.

1. Diagrama de casos de uso.

Casos de Uso

Los casos de uso son una técnica de modelización de requisitos funcionales que facilitan la comunicación entre los desarrolladores, los clientes y los usuarios finales del sistema. Su lenguaje sencillo es comprensible por todos los implicados en el proceso de desarrollo de un sistema software.

Los casos de uso, en su conjunto, describen los distintos usos que se le quiere dar al sistema.

Diagrama de casos de uso

El diagrama de casos de uso especifica el comportamiento global del sistema y su interacción con el entorno. Muestra los servicios o funciones del sistema y los roles de los elementos del entorno con los que interactúan. Por ejemplo, el rol de usuario de un sistema. A estos roles de los elementos del entorno se les denomina actores.

Cada servicio que el sistema deba realizar se modela como un caso de uso y cada rol de los elementos del entorno del sistema se modela como un actor.

En el diagrama de casos de uso, se delimitan las fronteras del sistema mediante una caja. Los elementos que queden fuera de la caja forman parte del entorno. Sus roles, es decir, los actores nunca son parte del sistema, aunque interactúen con él.

Los actores y los casos de uso se relacionan mediante asociaciones. Una relación de asociación entre un actor y un caso de uso indica que existe comunicación entre ellos y que pueden intercambiar información en ambos sentidos.

Es posible que el número de casos de uso que necesitemos para modelar un sistema sea demasiado grande para mostrarse en un único diagrama sin perder visibilidad y comprensibilidad. En ese caso, el diagrama se puede organizar agrupando los casos de uso en paquetes.

Especificación de un caso de uso

El comportamiento de un caso de uso se especifica describiendo la secuencia de acciones que el sistema debe llevar a cabo para proporcionar un servicio. Esta secuencia de acciones, habitualmente denominada flujo de eventos, debe escribirse de forma que sea lo suficientemente clara como para que alguien ajeno al sistema pueda entenderlo fácilmente.

2. Diagrama de clases.

Un diagrama de clases muestra las clases que componen el sistema y las relaciones que existen entre ellos. Este diagrama se utiliza para modelar la vista de diseño estructural de un sistema. Los diagramas de clases además, pueden contener paquetes.

Clases

Una clase es la definición de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

El número de instancias que pueden crearse de una clase es su multiplicidad. Generalmente la multiplicidad de una clase es ilimitada, en un sistema suele haber muchas instancias u objetos de una clase ejecutándose. Sin embargo, a veces es necesario restringir a un número determinado el número de instancias de una clase.

UML permite especificar dos características importantes de los elementos (atributos y

operaciones) de una clase: la visibilidad y el alcance.

Visibilidad**:** los elementos de una clase pueden ser públicos, protegidos o privados.

Alcance: se pueden especificar dos niveles de alcance, instancia (cada instancia de la clase tiene su propio valor del elemento) y clase (sólo hay un valor del elemento para todas las instancias de la clase).

Atributos

Relaciones

Una clase puede tener una relación consigo misma, indicando que los objetos de esa clase están conectados entre sí.

- Dependencia

Cuando objetos de una clase utilizan objetos de otra clase existe una relación de dependencia entre sus clases respectivas.

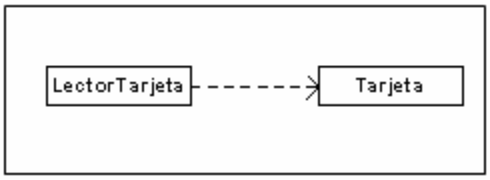


Ilustración 6 Dependencia entre clases

- Asociación

Una asociación es una relación estructural. Esta relación expresa que se puede navegar desde los objetos de una clase hasta los objetos de la otra clase.

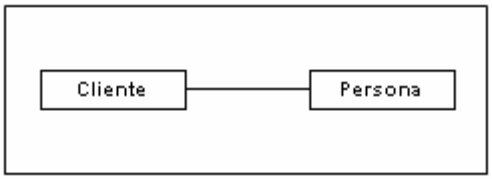


Ilustración 7 Asociación entre clases

Una asociación entre dos clases es una relación entre iguales, conceptualmente ninguna de las clases tiene más importancia que otra. Sin embargo, a veces conviene destacar que una de las clases es un "todo" del que la otra clase forma "parte". A esta relación "todo/partes" se le llama agregación simple. Gráficamente se representa con un rombo vacío en el extremo de la clase "todo".

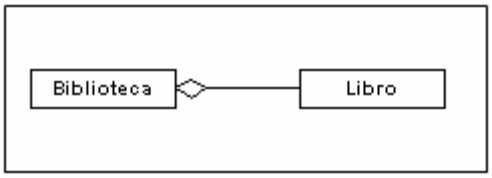


Ilustración 8 Agregación de clases

Existe otro tipo de agregación, la composición o agregación compuesta. La composición es también una relación "todo/partes", pero con fuertes implicaciones de comportamiento. La composición liga la existencia de las partes al todo: si el todo desaparece también desaparecen sus partes.

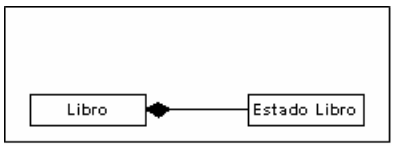


Ilustración 9 Agregación de clases "todo"

- Generalización

La generalización o herencia expresa una relación entre una clase genérica y una o varias clases específicas. A la clase genérica se le llama clase madre y a las clases específicas hijas. La generalización indica que las hijas heredan los atributos, operaciones y relaciones de la clase madre.

3. Diagrama de objetos.

Interfaces y realizaciones

A las clases abstractas puras, es decir, a las clases que no contienen ninguna

implementación, se les llama interfaces.

En UML una interfaz es una colección de operaciones que sirven para especificar los servicios de una clase o un componente. Una interfaz sólo contiene las cabeceras de las operaciones, no su implementación

Para que una interfaz se pueda usar hace falta que otra clase implemente las operaciones que la interfaz especifica. A esta relación entre la interfaz y la clase que la implementa se le llama realización. La realización indica que la clase implementa todas las operaciones de la interfaz.

Diagrama de objetos

Un diagrama de objetos muestra un conjunto de objetos y sus relaciones en un instante de tiempo determinado. Puede verse como una fotografía del sistema que muestra el estado de los objetos en ese instante.

La única relación entre objetos que se puede representar en UML es el enlace. Un enlace indica una conexión entre dos objetos. Dos objetos pueden estar conectados si existe una asociación o una dependencia entre las clases que instancian.

Los diagramas de objetos pueden contener paquetes y, cuando se quiere mostrar la clase que hay detrás de cada instancia, también pueden contener clases.

4. Diagrama de secuencias.

Los diagramas de interacción muestran comportamientos parciales del sistema, describiendo la secuencia de mensajes que intercambian los objetos para llevar a cabo una tarea.

En UML existen dos tipos de diagramas de interacción: los diagramas de secuencia y los diagramas de colaboración. Ambos son equivalentes, la diferencia entre ellos está en los aspectos que resaltan. Los diagramas de secuencia destacan el orden temporal de los mensajes, mientras que los diagramas de colaboración destacan la organización estructural de los objetos.

5. Diagrama de estados.

Un diagrama de estados modela la vida de un objeto mediante una máquina de

estados. Cada estado representa una situación durante la cual el objeto satisface alguna condición, realiza alguna actividad o espera algún evento. Se pueden definir dos estados especiales:

- *Estado inicial*: indica el punto de comienzo de la ejecución de la máquina de estados. Se representa con un círculo negro.
- *Estado final:* indica la terminación de la ejecución de la máquina de estados. Se representa con un círculo negro dentro de un círculo blanco.

6. Diagrama de actividades.

Los diagramas de actividades de UML son similares a los diagramas de flujo tradicionales. Generalmente se utilizan para modelar flujos de trabajo o para describir detalladamente una operación.

En UML los diagramas de actividades son un caso particular de los diagramas de estado que muestran un flujo de control. En estos diagramas los estados representan actividades o acciones. Una acción es una operación atómica indivisible que no puede ser interrumpida durante su ejecución. Una actividad es una operación no atómica que puede descomponerse en otras actividades o acciones y que puede ser interrumpida durante su ejecución.

7. Diagrama de componentes.

Un diagrama de componentes muestra la organización y las dependencias entre un conjunto de componentes. Los diagramas de componentes pueden contener paquetes para organizar los elementos.

8. Diagrama de despliegue.

Un diagrama de despliegue muestra la configuración de los nodos del sistema. En UML, un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional que, generalmente, tiene alguna memoria y, a menudo, capacidad de procesamiento.

2.3 La definición del problema

En la actualidad el número de catástrofes naturales o provocadas por el ser humano están aumentando, tanto a nivel nacional como internacional, y es necesario tener herramientas que nos ayuden a gestionar de la manera más rápida y eficaz posible estos hechos; entre los mismos, se pueden citar los sistemas ARCE y SIGAME. Como ya se explicó al inicio del capítulo en **2.1 El contexto del problema**, estos sistemas han sido concebidos para mejorar la gestión de medios y recursos, por lo que objetivamente podrían trabajar de forma coordinada; sin embargo, existen una serie de factores que impiden este hecho:

1. En primer lugar, tienen filosofías distintas, en el caso de SIGAME no existe un organismo central que obligue a la cesión de medios. Sin embargo, en ARCE lo utiliza una organización: la Asociación Iberoamericana de Organismos Gubernamentales de Defensa y Protección Civil. La diferencia es que en ARCE todos los organismos participantes (los países) tienen el mismo perfil; sin embargo, en SIGAME, la DGPCE tiene un perfil distinto que las CCAA. Las CCAA hacen solicitudes y realizan ofertas, la DGCPE comprueba que esas solicitudes y ofertas son correctas y completas, pero no gestiona porque las CCAA son independientes, sin embargo si intenta coordinar o introducir cierto control. Por tanto, SIGAME no puede utilizarse en la Asociación y ARCE no puede utilizarse en España.
2. Sus ámbitos de actuación difieren, mientras, SIGAME es utilizado para coordinar la respuesta de la DGPCE y las comunidades autónomas de España, ARCE es utilizado en países Iberoamericanos. Estos son ámbitos independientes que, sin embargo, podrían colaborar ante una emergencia. Por ejemplo, ante una emergencia sucedida en Colombia podría utilizar ARCE para solicitar medios, que esa incidencia llegará a todas las CCAA de España y que estás cedieran sus medios. El único organismo que tiene acceso a los dos sistemas es la DGPCE, con lo que se crea un cuello de

botella.

3. Y usan tecnologías diferentes, ARCE está desarrollado para ser ejecutado en ZOPE utilizando PostgreSQL; por otro lado, SIGAME se concibió como una aplicación J2EE utilizada sobre Tomcat y con MySQL como gestor de base de datos. Con lo cual, son sistemas con una misma arquitectura pero dispares tecnológicamente.

Por tanto, el único punto en común es el tipo de información que gestionan, por lo que la coordinación entre los mismos debe reducirse a este punto. Se trata de definir un estándar común que trabaje sobre la información, es decir, que compartiendo dicha información se colabore en la solución de una emergencia.

Esta gestión que ofrecen ambos proyectos trata de mantener la información ordenada, y sabiendo en cada momento que recursos son necesarios para solventar una emergencia, de que tipo y cuáles ya están ofertados, entre otros trámites. Para lograr este fin, es necesario coordinar eficientemente las actividades y recursos, así como compartir la información entre las distintas organizaciones involucradas en la resolución de la emergencia. Por este motivo es importante que interoperen o colaboren ambos sistemas de emergencia Web, y se pueda llegar a una recuperación rápida y sobretodo, eficaz, ante una situación de emergencia o catástrofe. Al traspasar información de una base de datos a otra existirá la posibilidad de aportar recursos a emergencias y recibir recursos de emergencias, que antes no estaban al alcance, y del mismo modo, se podrá informar del estado de las emergencias en cada momento.

3 Gestión de proyecto software

La gestión de proyectos [11] es la disciplina de organizar y administrar recursos de tal manera que se pueda culminar todo el trabajo requerido dentro del alcance, el tiempo y coste definido. La gestión de proyectos tiene como finalidad principal la planificación, el seguimiento, y el control de las actividades y recursos que intervienen en el desarrollo de un sistema de información. Como consecuencia de este control es posible conocer en todo momento que problemas se producen y resolverlos o paliarlos de manera inmediata.

Se tendrán en cuenta una serie de factores que condicionan la viabilidad del proyecto. En primer lugar se realizará una breve definición del proyecto, posteriormente, veremos la organización del proyecto donde se concretan las interfaces externas que emplearemos, se describen los participantes del proyecto, los roles y las responsabilidades que ejercerán. A continuación, se lleva a cabo un plan de trabajo donde se estudiarán las fases que se realizarán en la elaboración del proyecto. En el siguiente apartado veremos el seguimiento con las distintas gestiones a cumplir durante dicho proyecto.

3.1 Definición del Proyecto

El presente proyecto cubre la transformación e intercambio de información entre dos sistemas específicos de gestión de emergencias. Ante una situación de emergencia, los sistemas realizan una acción cuyo resultado queda reflejado en la base de datos. En este momento, se lanza un evento que inicia la recuperación, transformación e intercambio de la información entre los sistemas implicados. El presente proyecto se centra en este proceso de manipulación e intercambio de la información. Los aspectos relacionados con el funcionamiento de las aplicaciones de interés no es objeto de este proyecto, limitándonos a un estudio de sus características y funcionalidades más significativas.

3.2 Ciclo de vida

La metodología empleada para la realización del proyecto será RUP [12], es un proceso para el desarrollo de un proyecto de un software que define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto. Como tres características esenciales, **está**

dirigido por los Casos de Uso, que orientan el proyecto a la importancia para el usuario y lo que este quiere, **está centrado en la arquitectura**, que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden, y **es iterativo e incremental,** donde divide el proyecto en mini proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada. Como filosofía RUP maneja seis principios clave:

1. Adaptación del proceso. El proceso deberá adaptarse a las características propias de la organización. El tamaño del mismo, así como las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.
2. Balancear prioridades. Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.
3. Colaboración entre equipos. El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados, etc.
4. Demostrar valor iterativamente. Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.
5. Elevar el nivel de abstracción. Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o esquemas (frameworks) por nombrar algunos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.
6. Elevar el nivel de abstracción. Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o esquemas (frameworks) por nombrar algunos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.
7. Enfocarse en la calidad. El control de calidad no debe realizarse al final de

cada iteración, sino en todos los aspectos de la producción.

El ciclo de vida RUP

RUP divide el proceso en 4 fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

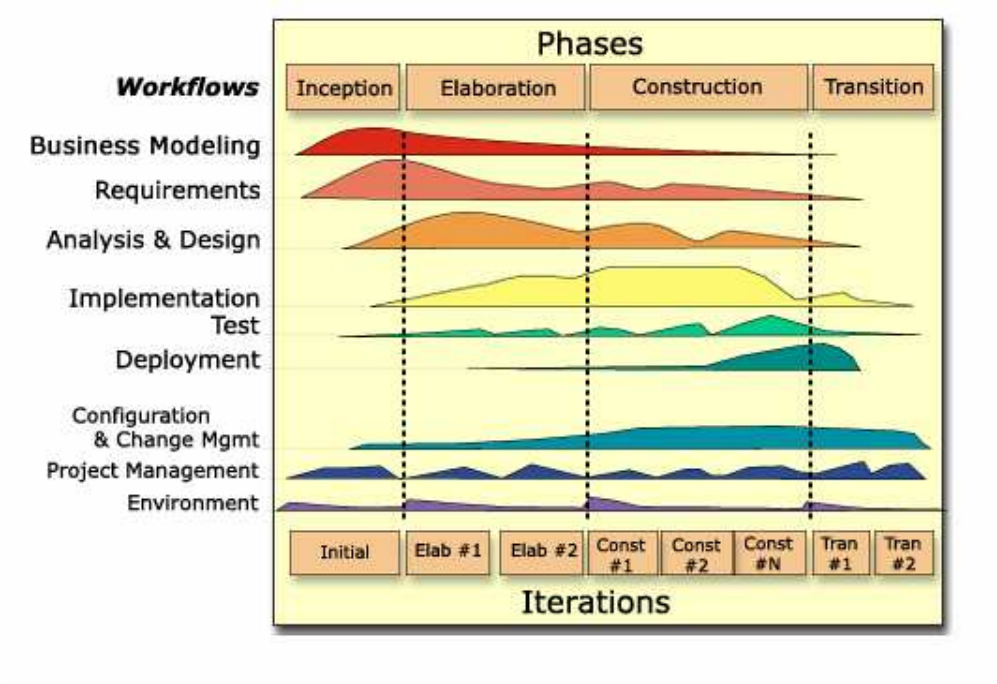


Ilustración 10 Fases del ciclo de vida de RUP

En las iteraciones de cada fase se hacen diferentes esfuerzos en diferentes actividades:

- Inicio: Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos. Se define el alcance del proyecto.
- Elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.
- Construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
- Transición: se Instala el producto en el cliente y se entrena a los usuarios.

Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

3.3 Organización del proyecto

En este apartado se definirán varias secciones donde se estudiará en primer lugar, las interfaces externas que tendremos en cuenta en el presente proyecto, luego la estructura interna de los responsables del proyecto y, por último, los roles que desempeñan estos responsables.

Participantes del proyecto

El propósito de Recursos Humanos y Ambiente de Trabajo es proporcionar los recursos humanos adecuados para cumplir las responsabilidades asignadas a los roles dentro de la organización, así como la evaluación del ambiente de trabajo. La estructura interna para poder afrontar el proyecto consiste en la exposición del RBS "humano". Tiene por objetivo representar la organización humana del proyecto, su estructura, y demás aspectos importantes en cuanto a su organización.

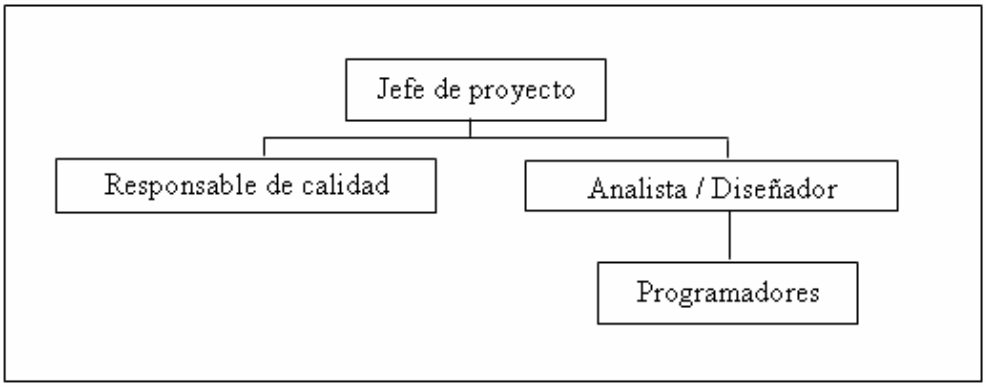


Ilustración 11 Estructura de los recursos humanos

Como se aprecia en la ilustración anterior dispondremos de dos áreas bien definidas e independientes como son: responsable de calidad (a tiempo parcial) y analista/diseñador (a tiempo completo). También podríamos incluir responsable de gestión de configuración, pero de las tareas que desempeñaría este perfil se encargará el analista/diseñador. Por

encima de ellos se situará el Jefe de proyecto (a tiempo parcial) y, aunque no está incluido, también influirá el departamento de recursos humanos. Dependiendo del analista (que también se encargará de las tareas de diseño) están los dos programadores, que se encomendarán a la tarea de implementación y colaborarán con el analista en algunas tareas de diseño, este recurso se empleará a tiempo total.

Roles y responsabilidades

En este apartado trataremos de diferenciar los roles existentes en los participantes del proyecto y las responsabilidades que tomarán. Así vemos en la siguiente tabla la descripción de las principales responsabilidades asociadas a cada uno de los puestos en el equipo de desarrollo, durante las fases de Inicio y Elaboración, de acuerdo con los roles que desempeñan en RUP. La siguiente tabla tendrá tres columnas, la primera identificará el puesto de trabajo a desempeñar, la segunda columna marcará las responsabilidades que toma el puesto, y en la tercera columna se mostrará la dedicación a la que se emplean los perfiles.

Puesto	Responsabilidad	Dedicación
Jefe de proyecto	Es el responsable máximo del proyecto. Asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. Además, el jefe de proyecto se encargará de supervisar el establecimiento de la arquitectura del sistema. Planificación y control del proyecto.	Se empleará a este proyecto a tiempo parcial.
Analista/Diseñador	Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos.	Se empleará a tiempo total.

Responsable de calidad	Se ocupará de asegurar la calidad de los productos resultantes y realizará un conjunto de actividades que servirán para: Reducir, eliminar y, lo más importante, prevenir las deficiencias de calidad de los productos a obtener. Además realizará la preparación de las pruebas funcionales.	El responsable de calidad se empleará en este proyecto a tiempo parcial.
Programador	Se ocuparán de la codificación de las funcionalidades diseñadas por los analistas, así como la colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario..	Se empleará a tiempo total.

Tabla 1 Roles y responsabilidades del proyecto

3.4 Fases del proyecto y estimación de tiempos

En cuanto a tipo el ciclo de vida de RUP, es una implementación del desarrollo en espiral, fue creado ensamblando los elementos en secuencias semi-ordenadas, El ciclo de vida organiza las tareas en fases e iteraciones, se descompone en 4 FASES secuenciales, cada cual concluye con un producto intermedio. Al terminar cada fase se realiza una evaluación para determinar si se ha cumplido o no con los objetivos de la misma.

Fases del ciclo de vida

1. Inicio (Inception)

   El objetivo general de esta fase es establecer un acuerdo entre todos los interesados acerca de los objetivos del proyecto. Esta fase es esencial para el desarrollo de nuevo software, ya que se asegura de identificar los riesgos relacionados con el negocio y requerimientos. Para proyectos de mejora de software existente esta fase es más breve y se centra en asegurar que vale la pena y es posible, desarrollar el proyecto.

2. Elaboración

   El objetivo en esta fase es establecer la arquitectura base del sistema a fin de proveer bases estables para el esfuerzo de diseño e implementación en la

siguiente fase. La arquitectura debe abarcar todas las consideraciones de mayor importancia de los requerimientos y una evaluación del riesgo.

3. Construcción

El objetivo de la fase de construcción es clarificar los requisitos y completar el desarrollo del sistema basados en la arquitectura base. Vista de cierta forma esta fase es un proceso de manufactura, en el cual el énfasis se torna hacia la administración de recursos y control de las operaciones para optimizar costos, tiempo y calidad.

4. Transición

Esta fase se enfoca en asegurar que el software este disponible para sus usuarios. Esta fase se puede subdividir en varias iteraciones, además incluye pruebas del producto para poder hacer el entregable del mismo, así como realizar ajuste menores de acuerdo a ajuste menores propuestos por el usuario. En este punto, la retroalimentación de los usuarios se centra en depurar el producto, configuraciones, instalación y aspectos sobre utilización.

Fase	Número de iteraciones	Duración
Fase de inicio	1	4 semanas
Fase de elaboración	1	2 semanas
Fase de construccion	2	4 semanas
Fase de transición	n/a	n/a

Tabla 2 Estimación de tiempos del proyecto

En la tabla anterior se pueden ver las fases del proyecto y su duración, también se refleja el número de iteraciones, en la fase de transición consideramos que no aplica puesto que el proyecto no se ha colocado a disposición de los usuarios. En la siguiente tabla vamos a ver los porcentajes de los aspectos de los casos de uso al acabar cada fase.

	Fase de concepción	Fase de elaboración	Fase de construcción	Fase de transición
Modelo de Negocio Terminado	50%-70%	Casi el 100%	100%	[no aplica]
Casos de uso identificados	50%	50% o más	100%	[no aplica]
Casos de uso descritos	10%	40%-80%	100%	[no aplica]
Casos de uso analizados	5%	20-40%	100%	[no aplica]
Casos de uso diseñados implementados y probados	Muy poco, puede que sólo algo relativo a un prototipo para probar conceptos	Menos del 10%	100%	[no aplica]

Tabla 3 Porcentajes de realización de fases

En la siguiente tabla (Tabla 4 Relación de fases y esfuerzos) podemos ver que los porcentajes de esfuerzo, de los tiempos dedicados y los participantes que colaboran en cada una de fases del proyecto. Podemos ver que al ser un proceso iterativo los participantes serán todos en todas las fases, sólo que no estarán dedicados el mismo tiempo.

	Inicio	Elaboración	Construcción	Transición
Esfuerzo	5%	20%	65%	10%
Participantes	Jefe de proyecto (20%) Analista/Diseñador (50%) Programadores (25%) Responsable de Calidad (5%)	Jefe de proyecto (15%) Analista/Diseñador (40%) Programadores (40%) Responsable de Calidad (5%)	Jefe de proyecto (10%) Analista/Diseñador (30%) Programadores (25%) Responsable de Calidad (5%)	Jefe de proyecto (10%) Analista/Diseñador (20%) Programadores (20%) Responsable de Calidad (50%)
Tiempo dedicado	10%	30%	50%	10%

Tabla 4 Relación de fases y esfuerzos

3.5 Seguimiento y Control del proyecto

El propósito del seguimiento del proyecto y su control es una actividad ubicada después de la planificación de proyecto y se realiza a lo largo de todo el ciclo de vida. Con ello se pretende facilitar una visión adecuada del progreso real, de forma que la dirección pueda tomar unas medidas eficaces cuando el desarrollo del proyecto software se desvía notablemente de los planes. Para ello es necesario establecer marcas de seguimiento. Esta acción la llevara a cabo el jefe de proyecto y los objetivos que se marcan son identificar diferencias entre lo planificado y lo realizado, evaluar el avance del proyecto, adaptar el plan de acción a las diferencias encontradas, prever desviaciones importantes, contribuir a la creación de históricos y contabilizar los costes de actividad.

Gestión de Requisitos

Los requisitos del sistema son especificados en el artefacto Visión. Cada requisito tendrá una serie de atributos tales como importancia, estado, iteración donde se implementa, etc. Estos atributos permitirán realizar un efectivo seguimiento de cada requisito. Los cambios en los requisitos serán gestionados mediante una Solicitud de Cambio, las cuales serán evaluadas y distribuidas para asegurar la integridad del sistema y el correcto proceso de gestión de configuración y cambios.

Control de Plazos

El calendario del proyecto tendrá un seguimiento y evaluación semanal por el jefe de proyecto y por el Comité de Seguimiento y Control.

Control de Calidad

Los defectos detectados en las revisiones y formalizados también en una Solicitud de Cambio tendrán un seguimiento para asegurar la conformidad respecto de la solución de dichas deficiencias Para la revisión de cada artefacto y su correspondiente garantía de calidad se utilizarán las guías de revisión y checklist (listas de verificación) incluidas en RUP.

Gestión de Riesgos

A partir de la fase de Inicio se mantendrá una lista de riesgos asociados al proyecto y

de las acciones establecidas como estrategia para mitigarlos o acciones de contingencia. Esta lista será evaluada al menos una vez en cada iteración.

Gestión de Configuración

Se realizará una gestión de configuración para llevar un registro de los artefactos generados y sus versiones. También se incluirá la gestión de las Solicitudes de Cambio y de las modificaciones que éstas produzcan, informando y publicando dichos cambios para que sean accesibles a todo los participantes en el proyecto. Al final de cada iteración se establecerá una baseline (un registro del estado de cada artefacto, estableciendo una versión), la cual podrá ser modificada sólo por una Solicitud de Cambio aprobada.

4 Solución

Este apartado expondrá las tareas realizadas y las decisiones tomadas en cada momento, así como las explicaciones oportunas, para llevar a cabo el desarrollo del proyecto. Este apartado está estructurado en tres bloques: el primero es la **Revisión de los sistemas**, donde se explicarán los dos sistemas en profundidad. Gracias a esta revisión se podrán conocer aquellas operativas relacionadas y la forma en la cual las acciones de un sistema influyen en el otro, veremos un primer apartado donde podremos observar los casos de uso de los sistemas, que nos llevan al desarrollo de nuestro proyecto, comenzará con los casos de uso del sistema ARCE y a continuación los del sistema SIGAME. En el segundo punto trataremos el modelo de datos de estos sistemas, es decir, veremos las estructuras de las bases de datos.

En el segundo bloque **Modelo de intercambio**, vamos a estudiar el modelo de información, éste a su vez estará estructurado en dos partes, la primera (**Modelo de información),** mostrará el modelo conceptual, es decir, se mostrarán las entidades que afectan al proyecto y las relaciones que existen entre dichas entidades. A continuación, veremos la estructura de las tablas que formarán el documento XML y con los tipos de datos que emplearán dichas tablas (**Diseño detallado**).

En el tercer bloque**4.3 Intercambio de la información**, este bloque a su vez está estructurado en tres apartados, el primero será el análisis, donde se estudian los requisitos del sistema, y se muestran los casos de uso que tendrá nuestro proyecto, y los diagramas de actividad y de secuencia de este. El siguiente apartado tratará el diseño donde veremos la forma en la que el proyecto está planteado, se distinguirá la arquitectura del sistema, la representación de la información, el tratamiento del XML y el envío y recepción del XML. El apartado tres tratará el despliegue que será los pasos a seguir para poder usar nuestro proyecto.

4.1 Revisión de los sistemas

En este apartado se pretende obtener una especificación de detallada de los sistemas para posteriormente construir una solución a la interoperabilidad, este primer bloque nos

servirá como base para ello.

Se trata de definir los sistemas tanto funcionalmente con los casos de uso como estructuralmente con las bases de datos. Debemos saber qué información existe en cada sistema y cuál será la forma de interactuar del actor con cada uno de ellos. En ARCE veremos que el actor puede realizar todos los casos de uso, sin embargo, en SIGAME existe un usuario superior que es la DGPCE, además de las comunidades autónomas.

El proceso de negocio

En este apartado estudiaremos los casos de uso de cada uno de los sistemas, tal y como vimos en Estado de la Cuestión (2.2), un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar como reacciona una respuesta a eventos que se producen en el mismo.

Comenzaremos con una descripción de los casos de uso del sistema ARCE, a continuación veremos los correspondientes a SIGAME.

Casos de uso ARCE

Como nombramos anteriormente, ARCE es un sistema Web diseñado para mejorar la respuesta conjunta ante situaciones de emergencia entre todos los miembros de la Asociación Iberoamericana de Organismos Gubernamentales de Defensa y Protección Civil que involucra a 21 países. El sistema proporciona mecanismos para notificar una emergencia, pedir recursos y ofrecer asistencia. Cuando una emergencia ocurre, el país afectado, utiliza ARCE para informar a la asociación sobre la situación. Para cada emergencia, la aplicación proporciona información actualizada sobre qué recursos son necesarios, las cantidades iniciales y cuántos elementos han sido ya facilitados, permitiendo así que cada asociado decida cómo contribuir.

A continuación, vemos el diagrama con los casos de uso (Ilustración 12), en este sistema solo existe un actor, que será el que denominaremos ARCE, puesto que todos los roles desempeñan los mismos casos de uso, no existe ningún actor específico que realice un caso diferente.

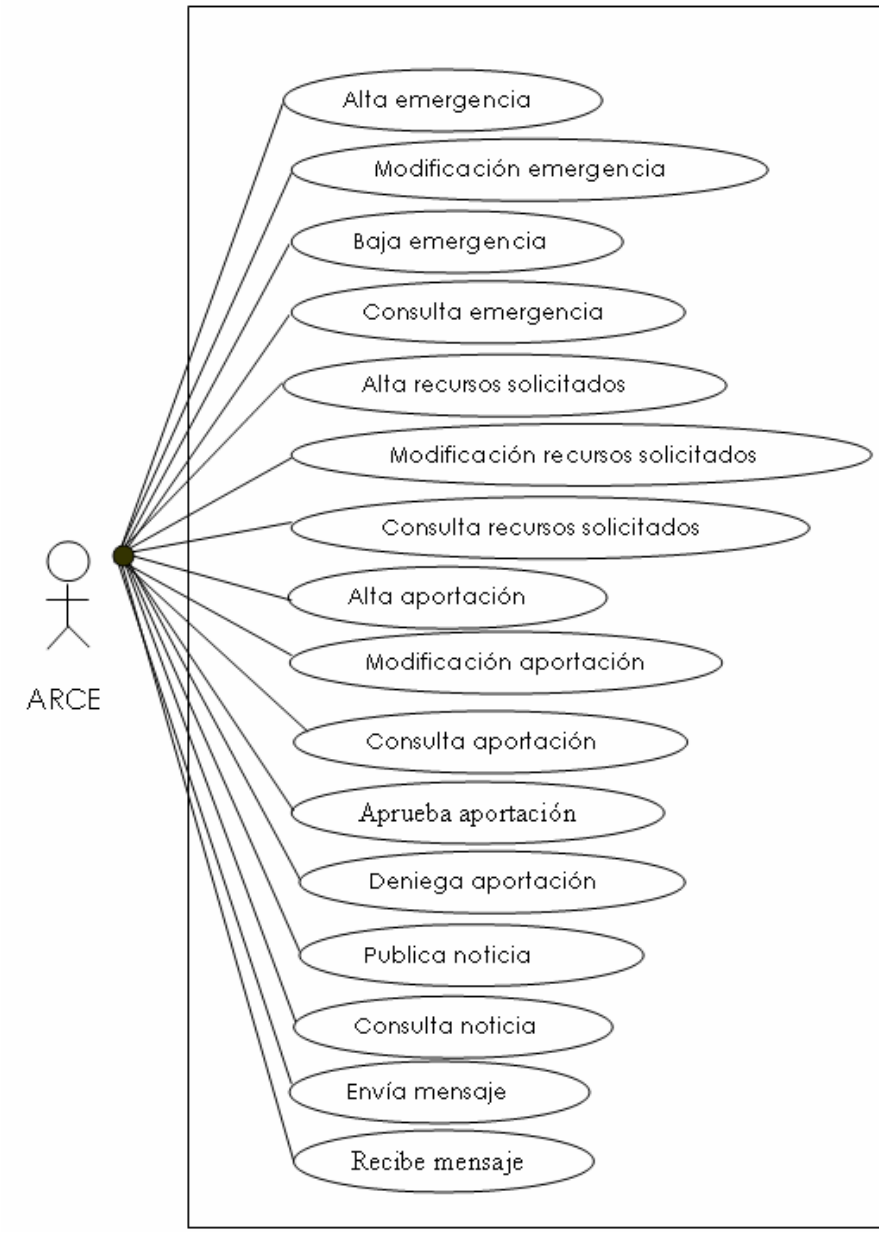


Ilustración 12 Casos de Uso del Sistema ARCE

De los casos de uso expuestos, a continuación en la Tabla 5 Casos de uso del sistema ARCE podemos distinguir dos estados, en situación de emergencia y en situación normal, al

primer estado se refieren los doce primeros casos de uso, y al estado de normalidad se refieren los cuatro siguientes. Los usuarios autorizados pueden publicar noticias que podrán ser leídas por cualquiera de los usuarios del sistema, e intercambiar mensajes entre ellos.

El código de los casos de uso de ARCE está formado de la siguiente manera:

- A → ARCE.
- SE→Situación de emergencia.
- SN→Situación normal.
- XX→Código numérico único.
- A + (SE | SN) + XX

Código	Nombre	Descripción
ASE01	Alta emergencia	Creación de un informe inicial de la situación de emergencia. Esta acción la realizará el país afectado.
ASE02	Modificar emergencia	Alterar el informe con el que anteriormente se ha dado de alta la emergencia en el sistema. Esta acción la realizará el país afectado.
ASE03	Baja emergencia	Se cambiará el estado de la emergencia en cuestión. Esta acción la realizará el país afectado.
ASE04	Consultar emergencia	Todos los países que constituyen ARCE podrán consultar las emergencias propias y las que han dado de alta otros países
ASE05	Alta solicitud de recursos	El país afectado puede realizar la solicitud detallada de los recursos necesarios.
ASE06	Modificar solicitud de recursos	El país afectado podrá modificar la solicitud de recursos.

ASE07	Consultar solicitud de recursos	Todos los usuarios del sistema de emergencia ARCE podrán realizar la consulta de los recursos solicitados por el país afectado.
ASE08	Alta aportación	Los países socios que previamente han realizado la consulta de recursos solicitados de los países afectados podrán ofertar medios movilizables.
ASE09	Modificar aportación	Los países aportantes pueden modificar su aportación antes de que esta sea aceptada o denegada.
ASE10	Consulta aportación	Los países podrán consultar la situación de las aportaciones formuladas por otros países.
ASE11	Aprueba aportación	El país afectado puede aceptar la aportación realizada por otros países.
ASE12	Deniega aportación	El país afectado puede rechazar la aportación realizada por otros países.
ASN01	Publica noticia	Publicar noticias que estén ocurriendo. Todos los países socios.
ASN02	Consulta noticia	Consultar las noticias existentes. Todos los países socios.
ASN03	Envía mensaje	Enviar mensajes a un país socio. Todos los países socios.
ASN04	Recibe mensaje	Recibir mensajes de un país socio. Todos los países socios.

Tabla 5 Casos de uso del sistema ARCE

Casos de uso SIGAME

En primer lugar, al igual que hicimos en el apartado anterior se mostrarán los casos de uso del sistema SIGAME. En SIGAME podemos mostrar dos actores, las comunidades autónomas y la Dirección General de Protección Civil Española (DGPCE), serán

identificados como “Comunidad autónoma” y “Usuario DGPCE” respectivamente. El usuario DGPCE puede realizar las funciones de las comunidades autónomas y además se encargará de decidir, organizar y coordinar las emergencias.

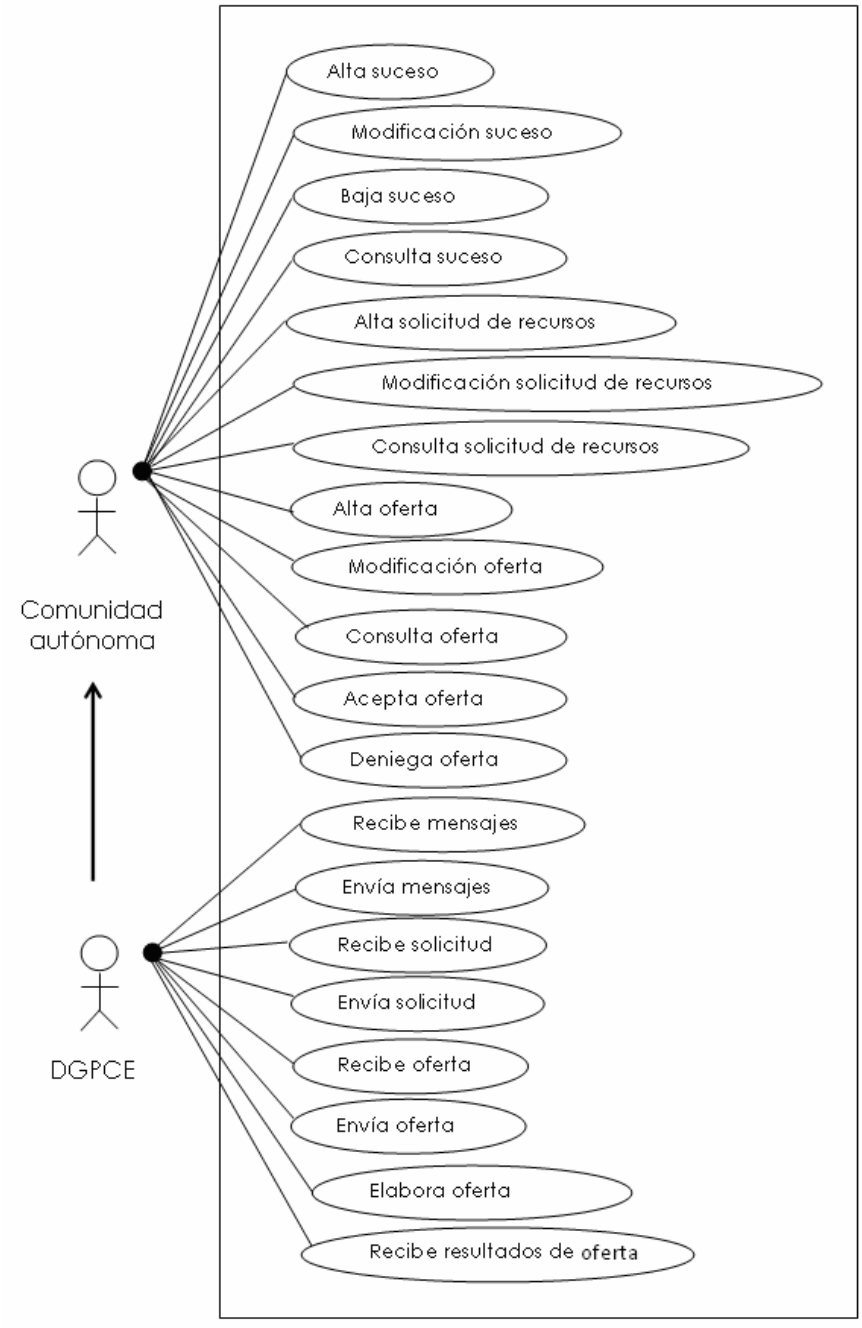


Ilustración 13 Casos de Uso del sistema SIGAME

A continuación, en la tabla se refleja una descripción de los casos de uso anteriores, el

código de los casos de uso de SIGAME están formados de la siguiente manera:

S → SIGAME

SE→Situación de emergencia.

XX→Código numérico único.

S + (SE) + XX

Código	Nombre	Descripción
SSE01	Alta suceso	La comunidad afectada rellena un formulario de alta en el sistema.
SSE02	Modificación suceso	La comunidad afectada modifica los datos de la solicitud que previamente ha creado.
SSE03	Baja suceso	La comunidad cierra la solicitud abierta, debe realizar esta acción la comunidad autónoma que abrió la solicitud o el usuario DGPCE.
SSE04	Consultar suceso	Cualquier usuario identificado en el sistema puede consultar las solicitudes creadas por otras comunidades autónomas o por si mismo.
SSE05	Alta solicitud de recursos	Cada solicitud de emergencia tiene una o varias peticiones de recursos que da de alta el usuario de comunidad autónoma afectada.
SSE06	Modificación solicitud de recursos	La comunidad afectada puede modificar la solicitud de recursos que anteriormente dio de alta.
SSE07	Consulta solicitud de recursos	Todas las comunidades pueden consultar los recursos solicitados por otras comunidades autónomas.
SSE08	Alta oferta	Las comunidades autónomas pueden ofertar recursos a las comunidades afectadas.

SSE09	Modificación oferta	Los usuarios identificados pueden modificar sus ofertas. Siempre que hayan sido dadas de alta por ellos mismos.
SSE10	Consulta oferta	Todos los usuarios pueden consultar las ofertas realizadas por otras comunidades y saber su estado.
SSE11	Acepta oferta	Las comunidades afectadas pueden aceptar las ofertas que han realizado las demás comunidades
SSE12	Deniega oferta	Las comunidades afectadas pueden denegar las ofertas que han realizado las demás comunidades
SSE13	Recibe mensajes	La DGPCE recibe mensajes.
SSE14	Envía mensajes	La DGPCE envía mensajes sobre las solicitudes para informar a las demás comunidades.
SSE15	Recibe solicitud	La DGPCE recibe las solicitudes y decide si se publican o no, es la primera que ve las emergencias.
SSE16	Envía solicitud	La DGPCE envía las solicitudes para informar a las demás comunidades.
SSE17	Recibe oferta	La DGPCE recibe las ofertas que otras comunidades hacen a las comunidades afectadas.
SSE18	Elabora oferta global	La DGPCE decide y elabora una oferta global que luego debe enviar a la comunidad afectada.
SSE19	Recibe resultados de oferta	Las comunidades afectadas envían a la DGPCE la decisión de aceptar o denegar las ofertas.

Tabla 6 Casos de uso del sistema SIGAME

El modelo de datos

En primer lugar veremos el estudio de SIGAME, las tablas que nos interesan de la Base de Datos son seis activamente, es decir, al dar de alta una emergencia, o una oferta se modificarán las siguientes tablas:

Código	Nombre de tabla	Descripción	Uso
SIG01	Solicitud	Se encarga de almacenar datos referentes a la "Solicitud", es decir, los campos que se llenarán serán los que faciliten información sobre la fecha en la que se da de alta la emergencia, descripción, lugar,…	[No aplica]
SIG02	Alertas_solicitudes	Tabla que recopila datos referentes a las correspondencias entre las solicitudes y las comunidades autónomas, teniendo cada uno un estado.	Al insertar una emergencia, es decir, una solicitud, se incluirán en esta tabla 20 tuplas, una por cada comunidad autónoma y otra por la DGPCE, asociadas al identificador de la solicitud que se acaba de insertar. El estado toma el valor "Normal" para todas las comunidades autónomas y "Actualizada" en el caso de la DGPCE. Significa que la solicitud puede ser visitada por todas las comunidades.

SIG03	Estado_evolucion	Tabla que recopila datos referentes al progreso de la emergencia.	La tabla estado_evolucion estará asociada a una solicitud, y contendrá campos más concretos sobre la emergencia como pueden ser número de fallecidos, desaparecidos…, en una fecha dada.
SIG04	Recursos_solicitados	Almacenará los recursos necesarios para solventar una emergencia. Los campos que almacena son por ejemplo, el nombre del recurso, la fecha en la que se da de alta esa petición de recurso,…	Cada solicitud de emergencia puede necesitar recursos. Estos recursos deben existir en la tabla recurso y en el catálogo de recursos.

SIG05	Oferta	Guarda los datos que tratan de las ofertas que se realizarán para cada solicitud en caso de emergencia.	Una oferta en SIGAME responde a un recurso solicitado, es decir, se supone que no se podrá ofertar un recurso que antes no haya sido solicitado, la fecha de la oferta se puede ampliar, y guardará datos de las unidades que ofertan, el estado del recurso, el destinatario, el contacto de la solicitud,… Actualmente, esta tabla almacena la información que en un principio se guardaba en la tabla "recursos_cedidos", es decir, una oferta se corresponderá por un recurso_cedido si el campo "estado" de la tabla oferta es "cedido", en caso de que sea "temporal" o "global" significará que ese recurso aun no se ha aceptado como oferta.
SIG06	recurso	Esta tabla guarda una descripción concreta de cada recurso, el nombre, las unidades, el titular, si está disponible, ofertado,… en cada momento.	Cada recurso debe existir en el catalogo de recursos.
SIG07	Catalogo_Recursos	Guarda información sobre los recursos.	Este catálogo debe ser común en SIGAME y en ARCE.

Tabla 7 Tablas de la base de datos SIGAME

En SIGAME las tablas vistas anteriormente se relacionan tal y como vemos en el diagrama de entidad relación de la siguiente imagen (Ilustración 14 Diagrama Entidad/Relación de la Base de Datos SIGAME), solo se muestran las claves primarias con el fin de clarificar la información.

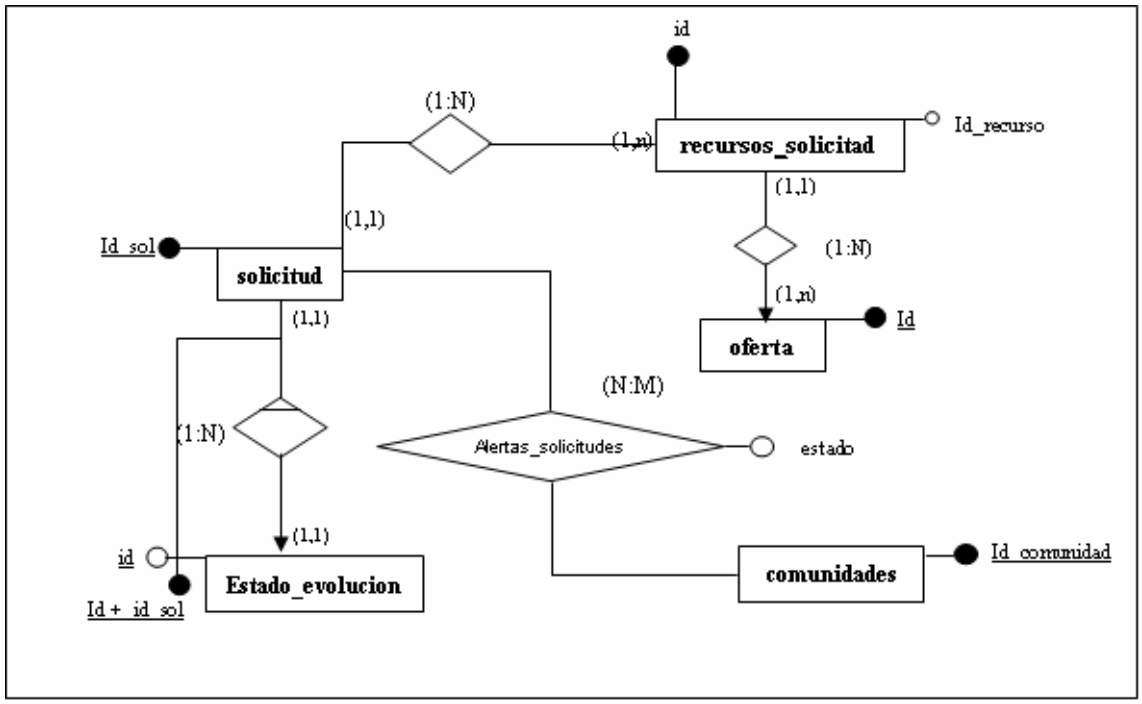


Ilustración 14 Diagrama Entidad/Relación de la Base de Datos SIGAME

En ARCE al dar de alta una emergencia, o insertar aportaciones se rellenan una serie de formularios que recogen los datos y los almacenan en unas tablas que se denominan de "primera_solicitud" son una primera aproximación de los datos de la emergencia, son para hacerse a la idea de lo que está ocurriendo durante las primeras 48 horas, una vez pasada esta aproximación los datos son trasladados a las tablas que utilizamos en este proyecto, el motivo de no usar las tablas de "primera_solicitud", es que al ser emergencias que provienen del sistema de emergencia SIGAME, estos datos ya son reales.

Las tablas que usaremos en el proyecto serán las citadas a continuación, como en el caso anterior existen tablas que sólo utilizaremos a modo de consulta:

Código	Nombre de Tabla	Descripción	Uso
ARC01	Emergencia	Almacena los datos concretos de una emergencia. En esta tabla se puede ver la evolución que va tomando una emergencia	Son datos reales, en ARCE existen tablas que guardan los datos orientativos que se corresponden con la información de las primeras 48 horas de la emergencia, en este caso la tabla se llamaría "primera_solicitud". Sabemos su evolución porque esta tabla posee un campo traza, el valor más alto de este campo es el que marca la última modificación, se guardarán datos como los daños causados por cada suceso.
ARC02	Solicitud	Almacena los datos principales de una emergencia, puesto que es una solicitud.	Esta tabla está asociada a la tabla Emergencia, se centra en el alta de una solicitud de una emergencia concreta, en una fecha, por un usuario… para que quede constancia.
ARC03	Recursos_solicitados	Es una tabla que guarda los datos asociados a una solicitud para un recurso concreto y una cantidad.	El recurso solicitado debe existir en el catalogo de recursos.
ARC04	Aportacion	Serán las aportaciones que ofrecen los demás países al país afectado.	Recopila datos sobre una aportación, que se realiza a una solicitud concreta, sabiendo la fecha, el estado del emisor,…

ARC05	Aportacion_cursada	Esta tabla se encarga de almacenar los datos de las aportaciones cursadas, datos como las fechas de inicio y fin de la aportación, el país,…	Insertaremos en esta tabla cuando introduzcamos una tupla en la tabla aportación.
ARC06	Recursos_aportados	Al igual que los recursos_solicitados, solo guarda el identificador del recurso, el código de la solicitud a la que pertenece y una cantidad. Marca los recursos que han sido aportados ya.	El recurso solicitado debe existir en el catalogo de recursos.
ARC07	Lugar_entrega	Es la que almacena para cada solicitud, el lugar donde se producirá la entrega de los recursos aportados.	El código de lugar debe existir en la tabla lugar y código de la solicitud debe existir en la tabla solicitud.

ARC08	Cod_lugar	Esta tabla almacena el código de lugar de los lugares que se encuentran en la jerarquía, donde se muestra que el lugar puede ser aeropuerto, área, estación o puerto.	En ARCE, es importante saber el lugar de entrega en cada momento, sin embargo, en SIGAME no se le da tanta importancia, esto será estudiado más adelante. Esta tabla muestra una jerarquía, es decir, existirá para cada solicitud uno o varios lugares de entrega pero siempre será un aeropuerto, o un área, o una estación, o un puerto.
ARC09	Aeropuerto	Guarda los aeropuertos de los países de la base de datos.	
ARC010	Area	Guarda las áreas de los países de la base de datos.	
ARC011	Estacion	Guarda las estaciones de tren de los países de la base de datos.	
ARC012	Puerto	Guarda los puertos de los países de la base de datos.	
ARC013	Roles	Almacena el login de los usuarios y su rol correspondiente.	Esta tabla la usaremos como consulta, para comprobar los roles de los usuarios puesto que sólo los usuarios de nivel 4 serán los que den de alta las emergencias que se migrarán a SIGAME.

ARC014	Usuario	Recoge todos los datos referentes a los usuarios.	Esta tabla también se usará para realizar las comprobaciones oportunas a la hora de migrar la información.
ARC015	Entidad	Recoge todos los datos referentes a las entidades.	Esta tabla también se usará para realizar las comprobaciones oportunas a la hora de migrar la información.

Tabla 7 Tablas empleadas de ARCE

El modelo relacional de las tablas que empleamos para la realización del proyecto de la base de datos del sistema de gestión de emergencias ARCE se representa en la siguiente imagen, en la cual solo mostramos las claves primarias con el motivo de ver lo más esencial.

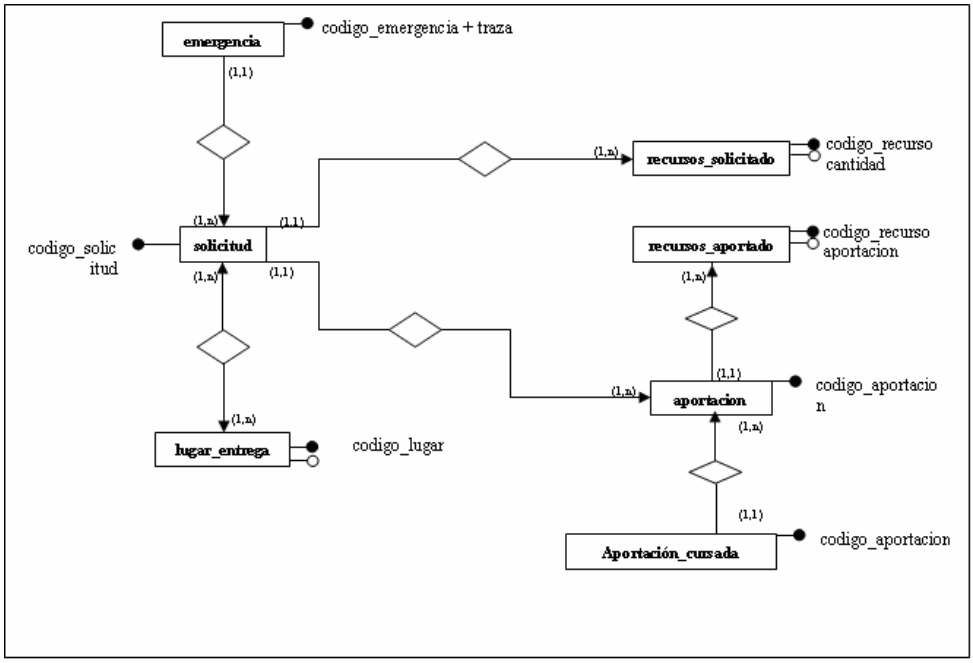


Ilustración 15 Diagrama Entidad/Relación de la base de datos (ARCE)

A continuación mostraremos la jerarquía de los lugares (no se incluirán las claves primarias):

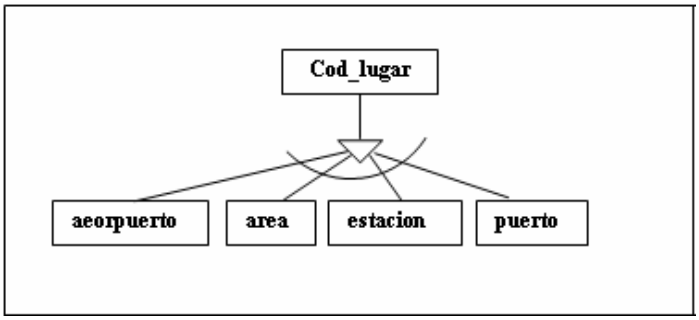


Ilustración 16 Jerarquía de los lugares de ARCE

4.2 Modelo de intercambio

En este apartado se pretende obtener una especificación de detallada del sistema que posteriormente se va a construir, y nos servirá como base para ello.

Se trata de definir cómo se debe llevar a cabo la transformación de datos, para ello lo primero que necesitábamos saber es qué información existe en cada sistema que es lo que trataremos en este punto y cuál es la forma de interactuar del actor con cada uno de los sistemas, que es lo que se estudió en el bloque anterior, 4.1 Revisión de los sistemas.

Modelo de información

Un modelo conceptual explica los conceptos más significativos en un dominio del problema, identificando los atributos y las asociaciones. Los casos de uso son una importante herramienta para el análisis de requerimientos, pero realmente no están *orientados a objetos*. Un modelo conceptual representa cosas del mundo real, no componentes del software. En UML se representa mediante un grupo de *diagramas de estructura estática* donde no se define ninguna operación. En estos diagramas se muestran conceptos (objetos), asociaciones entre conceptos (relaciones) y atributos de conceptos (atributos).

La siguiente figura (Ilustración 18 Modelo conceptual del proyecto) muestra el modelo conceptual del proyecto, en el cual se exponen las nociones de emergencia, solicitud, ofertas y lugares, y las relaciones que existen entre ellos.

Debemos separar los conceptos y entidades, por una parte un organismo será el encargado de informar de una emergencia, la cual tendrá victimas y sucederá en un lugar, por otro lado, este organismo realiza la solicitud de medios, la cual contendrá el lugar donde se realizará la entrega y los medios necesarios, los demás usuarios ofertarán estos medios, es decir, podrá tener una o varias ofertas asociadas.

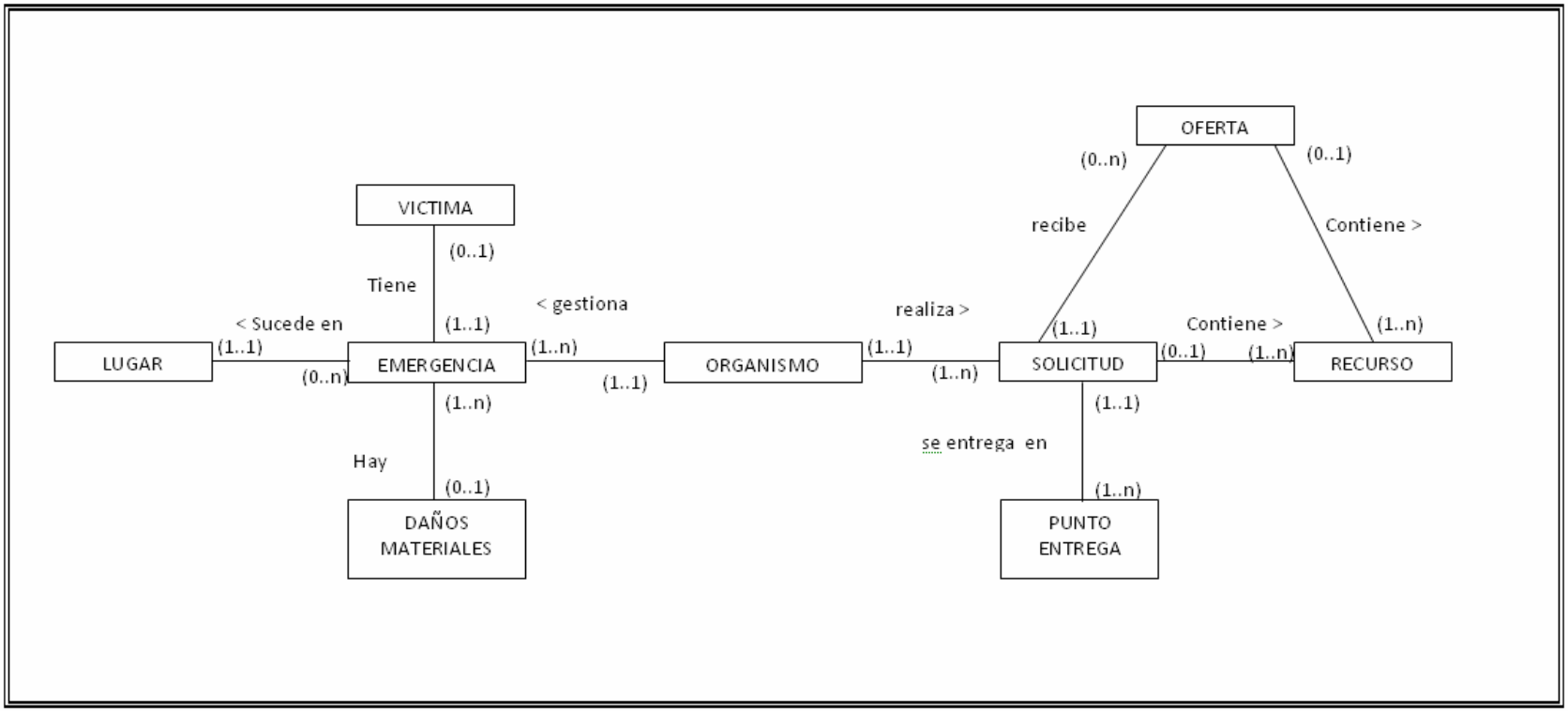


Ilustración 17 Modelo conceptual del Proyecto

En la imagen anterior (ilustración 18 Modelo conceptual del proyecto) hemos expuesto el modelo conceptual, formado por entidades necesarias y genéricas para cualquier tipo de emergencia.

Las entidades que hemos visto en el modelo conceptual, se relacionarán del mismo modo en el diagrama de clases, estas entidades se refieren a los datos de las emergencias, para una **emergencia** se tendrá en cuenta que podrán existir o no **victimas** y/o **daños materiales**, estas entidades se pueden ver en nuestro proyecto como una descripción de los daños sufridos a causa del suceso, es decir, si existen solo existirá uno para cada emergencia.

La emergencia ocurre siempre en un **lugar**, y en un lugar se pueden producir varias emergencias. La entidad lugar guardará los datos necesarios que describen un lugar concreto, contendrá los atributos latitud y longitud entre otros.

La emergencia siempre será gestionada por un **organismo** aunque este organismo puede gestionar varias emergencias, será el organismo el que realiza las solicitudes asociadas a las emergencias. Cada una de las **solicitudes** que realice el organismo se entregarán en un lugar de entrega, este punto de entrega solo se corresponderá con una solicitud.

Cada solicitud contiene **recursos**, estos recursos serán los que se encontrarán a su vez ofertados, esto se refleja en la entidad **oferta.** Las solicitudes recibirán o no estas ofertas.

Diseño detallado

En este apartado se realizará un estudio detallado del diseño de la solución, veremos la Representación de la Información mediante el diagrama de clases que define los sistemas, para llevar a cabo la solución debemos recordar que lo que se pretende funcionalmente es pasar información concreta de un sistema a otro, para ello la información se captará de un sistema y se almacenará en un fichero antes de insertarla en el sistema contrario, este fichero será un documento XML. El tratamiento del XML que recopilará la información se verá en el segundo punto de este apartado, Tratamiento del XML. En el último apartado (El envío y recepción del XML) se describirá la forma en que se envía y recepciona este XML que contiene los datos, es decir, el

diseño de las clases que nos permitirán enviar y recibir los ficheros mediante FTP.

Representación de la información

En el apartado anterior Modelo de Información, vimos el modelo conceptual del proyecto, donde se podían ver las relaciones entre las entidades que formarán parte de la solución, para recordarlo en la siguiente tabla veremos las relaciones que existen entre las entidades y las cardinalidades entre clases.

Entidad	Relación	Entidad relacionada	Cardinalidad
Emergencia	Hay	Victimas	0..1
Emergencia	Tiene	Daños_materiales	0..1
Emergencia	Sucede_en	Lugar	1
Emergencia	Es_gestionada_por	Organismo	1
Victimas	Pertenecen_a	Emergencia	1
Daños_materiales	Pertenecen_a	Emergencia	1
Lugar	Pertenece_a	Emergencia	0..*
Organismo	Gestiona	Emergencia	1..*
Organismo	Realiza	Solicitud	1..*
Solicitud	Recibe	Oferta	0..*
Solicitud	Se_entrega_en	Punto_entrega	1..*

Solicitud	Contiene	Recurso	1..*
Solicitud	Es_realizada_por	Organismo	1
Punto_entrega	Pertenece_a	Solicitud	1
Oferta	Contiene	Recurso	1..*
Oferta	Es_gestionada_por	Organismo	1
Recurso	Pertenece_a	Solicitud	0..1
Recurso	Pertenece_a	Oferta	0..1

Tabla 8 Relaciones entre entidades

En la siguiente ilustración veremos el diagrama de clases correspondiente al proyecto, en el cual se describe la estructura de la solución mostrando sus clases, atributos y las relaciones entre ellos.

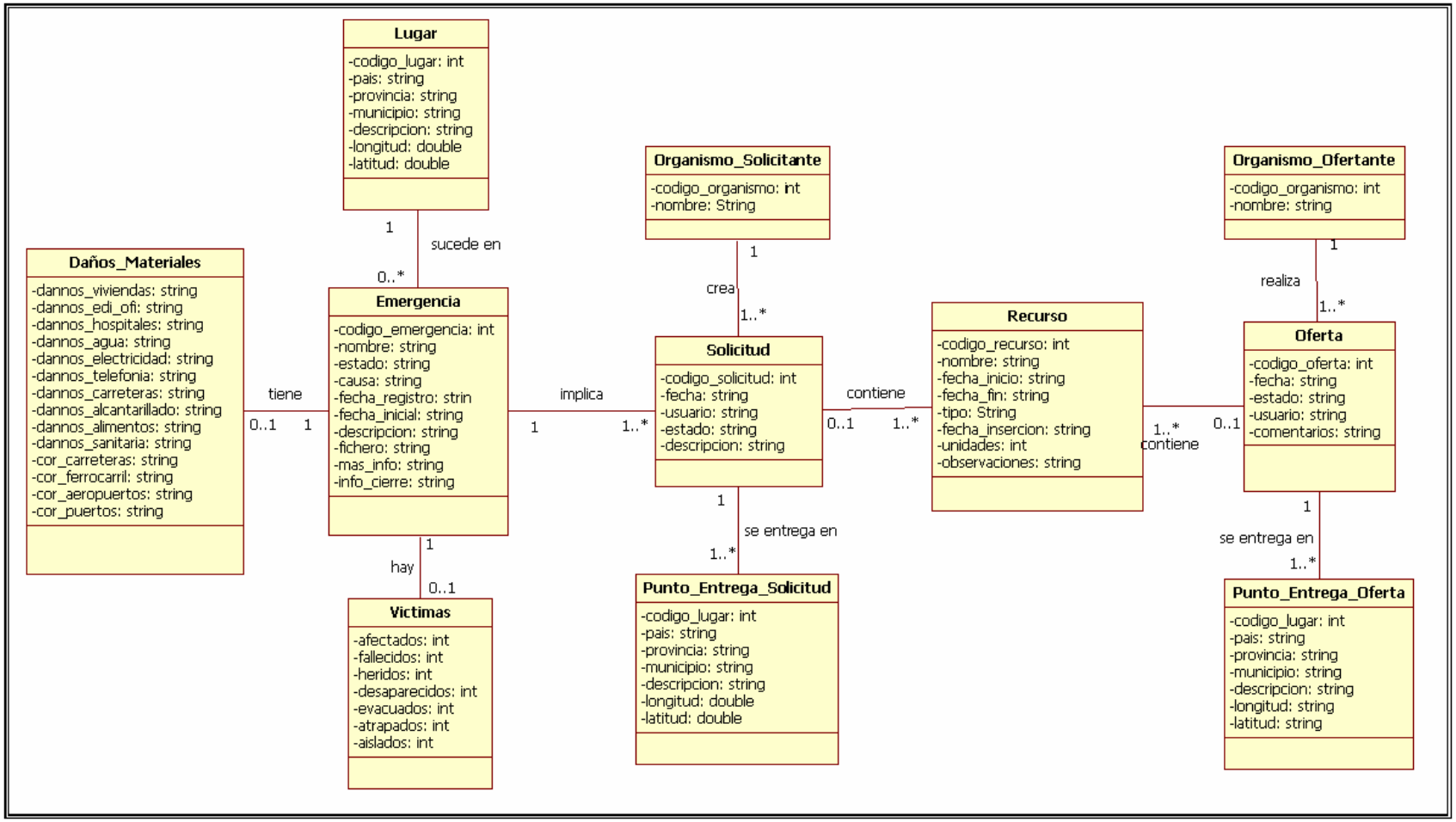


Ilustración 18 Diagrama de clases

En la imagen anterior vemos el diagrama de clases del proyecto, podemos observar que cambia con respecto al modelo conceptual, la estructura es la misma solo que se consideró que cada solicitud debía tener un lugar de entrega con el motivo de ser mas general y reusable, por el mismo motivo también se le asigna un lugar de entrega a las ofertas. El organismo que proponíamos como único para la solicitud de medios, se considera que debe existir un organismo que realice la solicitud y otro que realice la oferta, puesto que no serán el mismo. Finalmente, el recurso se entenderá como que una solicitud contiene recursos y ese recurso en concreto puede tener asociada una oferta o no.

En el ANEXO B podremos ver la representación de datos que almacenará el esquema XML que hemos definido acorde con el diagrama de clases visto en la página anterior. Podremos ver como una emergencia de tipo Temergencia guarda todos los datos referentes a dicha emergencia, las solicitudes con su contenido de recursos…; En el fichero podrán existir varias emergencias, dentro del elemento emergencias_pfc.

A continuación, vamos a exponer qué atributos contendrá cada entidad anteriormente mencionada, las tablas contendrán cinco columnas, la primera de ellas se refiere al atributo que posee la entidad, la siguiente será el tipo de datos que almacenará dicho atributo; La columna Nulo expondrá si un atributo puede o no ser nulo. La siguiente columna “valor por defecto” se pondrá el valor inicial que van a tener los atributos, por último, también se incluye una columna que guardará la descripción de los atributos.

El orden será el siguiente:

1. Entidad Emergencia
2. Entidad Victimas
3. Entidad Lugar
4. Entidad DañosMateriales
5. Entidad Organismo_solicitante
6. Entidad Organismo_ofertante
7. Entidad Solicitud
8. Entidad PuntoEntregaSolicitud
9. Entidad PuntoEntregaOferta
10. Entidad Recurso

11. Entidad Oferta

Entidad Emergencia

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_emergencia	Int (11)	No	--	--	Identificador único de la emergencia.
Nombre	Varchar(60)	No	--	--	Nombre por el que se conoce a la emergencia.
Estado	Varchar(1)	No	'A' , 'C'*	'A'	Estado en que se encuentra la emergencia.
Causa	Varchar(60)	No	--	--	Causas que hicieron que se produjera.
Fecha_registro	Varchar(19)	No	--	--	Fecha de registro en el sistema.
Fecha_inicial	Varchar(19)	No	--	--	Fecha de inicio de la emergencia.
Descripcion	Longtext	Si	--	--	Detalles de la emergencia.
Fichero	Varchar(100)	Si	--	--	
Mas_info	Longtext	Si	--	--	Detalles de la emergencia.
Info_cierre	Longtext	no	--	--	Mensaje de cierre, es nula hasta que la emergencia se solventa.

Tabla 9 Entidad Emergencia

*A→ abierta; C → cerrada

Entidad Victimas

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Afectados	Int (11)	No	--	0	Número de personas afectadas por el suceso.
Fallecidos	Int (11)	No	--	0	Número de personas fallecidas por el suceso.
Heridos	Int (11)	No	--	0	Número de personas heridas por el suceso.
Desaparecidos	Int (11)	No	--	0	Número de personas desaparecidas por el suceso.
Evacuados	Int (11)	No	--	0	Número de personas evacuadas por el suceso.
Atrapados	Int (11)	No	--	0	Número de personas atrapadas por el suceso.
Aislados	Int (11)	No	--	0	Número de personas aisladas por el suceso.

Tabla 10 Entidad Vìctimas

Entidad Lugar

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_lugar	Int(11)	No	--	--	Identificador único del lugar.
Pais	Varchar(50)	No	--	--	País en el que ocurre el suceso.

Provincia	Varchar(50)	Si	--	--	Provincias en el que ocurre el suceso.
Municipio	Varchar(50)	Si	--	--	Municipio en el que ocurre el suceso.
Descripcion	Varchar(200)	Si	--	--	Descripción del lugar.
Longitud	Double	Si	--	--	Coordenada donde ocurre el suceso.
Latitud	Double	si	--	--	Coordenada donde ocurre el suceso

Tabla 11 Entidad Lugar

Daños materiales

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Dannos_viviendas	Varchar(200)	No	--	--	Daños producidos en las viviendas por el suceso.
Dannos_edi_ofi	Varchar(200)	No	--	--	Daños producidos en los edificios oficiales por el suceso.
Dannos_hospitales	Varchar(200)	No	--	--	Daños producidos en los hospitales por el suceso.
Dannos_agua	Varchar(200)	No	--	--	Daños producidos en el suministro de agua por el suceso.
Dannos_electricidad	Varchar(200)	No	--	--	Daños producidos en el suministro eléctrico por el suceso.
Dannos_telefonia	Varchar(200)	No	--	--	Daños producidos en el suministro telefónico por el suceso.

Dannos_carreteras	Varchar(200)	No	--	--	Daños producidos en las carreteras por el suceso.
Dannos_alcantarillado	Varchar(200)	No	--	--	Daños producidos en el alcantarillado por el suceso.
Dannos_alimentos	Varchar(200)	No	--	--	Daños producidos en las materias alimentarias por el suceso.
Dannos_sanitaria	Varchar(200)	No	--	--	Daños producidos en la sanidad por el suceso.
Cor_carreteras	Varchar(200)	No	--	--	Cortes producidos en las carreteras por el suceso.
Cor_ferrocarril	Varchar(200)	No	--	--	Cortes producidos en el transporte ferroviario por el suceso.
Cor_aeropuertos	Varchar(200)	No	--	--	Cortes producidos en el transporte aéreo por el suceso.
Cor_puertos	Varchar(200)	No	--	--	Cortes producidos en el transporte portuario por el suceso.

Tabla 12 Entidad Daños materiales

Entidad Organismo_Solicitante

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_organismo	Int(11)	No	--	--	Identificador único del organismo.
Nombre	Varchar(60)	No	--	--	Nombre del organismo

Tabla 13 Entidad organismo_ofertante

Entidad Organismo_Ofertante

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_organismo	Int(11)	No	--	--	Identificador único del organismo.
Nombre	Varchar(60)	No	--	--	Nombre del organismo

Tabla 14 Entidad organismo_solicitante

Entidad Solicitud

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_solicitud	Int(11)	No	--	--	Identificador único de la solicitud.
Fecha_solicitud	Varchar(19)	No	--	--	Fecha en que se inserta la solicitud.
Usuario	Varchar(50)	Si	--	--	Usuario que inserta la solicitud.
Estado	Varchar(10)	Si	'P','M' *	'P'	Estado en que se encuentra la solicitud
Descripcion	Varchar(200)	Si	--	--	Descripción de la solicitud.

Tabla 15 Entidad Solicitud

*P→ pendiente; M → modificada

Entidad Punto_Entrega_Solicitud

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_punto	Int(11)	No	--	--	Identificador único del lugar.
Nombre	Varchar(50)	No	--	--	Nombre del punto de entrega.
Pais	Varchar(50)	Si	--	--	País donde se realizará la entrega.
Descripcion	Varchar(100)	Si	--	--	Descripcion del lugar donde se realizará la entrega
Longitud	Double	Si	--	--	Coordenada donde ocurre el suceso.
Latitud	Double	si	--	--	Coordenada donde ocurre el suceso

Tabla 16 Entidad punto_entrega_solicitud

Entidad Punto_Entrega_Oferta

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_punto	Int(11)	No	--	--	Identificador único del lugar.
Nombre	Varchar(50)	No	--	--	Nombre del punto de entrega.
Pais	Varchar(50)	Si	--	--	País donde se realizará la entrega.

Descripcion	Varchar(100)	Si	--	--	Descripcion del lugar donde se realizará la entrega
Longitud	Double	Si	--	--	Coordenada donde ocurre el suceso.
Latitud	Double	si	--	--	Coordenada donde ocurre el suceso

Tabla 17 Entidad punto_entrega_oferta

Entidad Recurso

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_recurso	Int(11)	No	--	--	Identificador único del recurso solicitado.
Nombre	Varchar(50)	No	--	--	Nombre del recurso solicitado.
Fecha_inicio	Varchar(19)	No	--	--	Fecha en que inicia la necesidad.
Fecha_fin	Varchar(19)	Si	--	--	Fecha en que acaba la necesidad.
Fecha_insercion	Varchar(19)	No	--	--	Fecha en que se inserta la solicitud del recurso.
Unidades	Int (10)	No	--	0	Numero de unidades que se solicitan.
Observaciones	Varchar(150)	si	--	--	Coordenada donde ocurre el suceso

Tabla 18 Entidad Recurso

Entidad Oferta

Atributo	Tipo de datos	Nulo	Valores	Valor por defecto	Descripción
Codigo_oferta	Int(11)	No	--	--	Identificador único de la oferta.
Fecha	Varchar(19)	No	--	--	Fecha en que se inicia la oferta.
Estado	Varchar(10)	No	'P','C' *	'P'	Estado en que se encuentra la oferta
Usuario	Varchar(20))	No	--	--	Usuario que inserta la oferta
Lugar_entrega	Int(11)	No	--	--	Lugar de la entrega.
Comentarios	Varchar(150)	si	--	--	Detalles de la oferta.

Tabla 19 Entidad Oferta

*P→ pendiente; C → cedido

Tratamiento del XML

En este apartado se explicará el diagrama de clases correspondiente a la lectura, escritura y envío del fichero. En este capítulo estudiaremos cómo se estructura la implementación, llevada a cabo sobre las decisiones tomadas en la fase de diseño.

En las siguientes imágenes podemos ver el diagrama de clases del código fuente, es decir la estructura de los paquetes y las clases de define de la siguiente manera, se ha partido la imagen para poder observar en detalle cada parte, existen tres partes, en la primera se puede ver la el módulo que se encarga de la lectura:

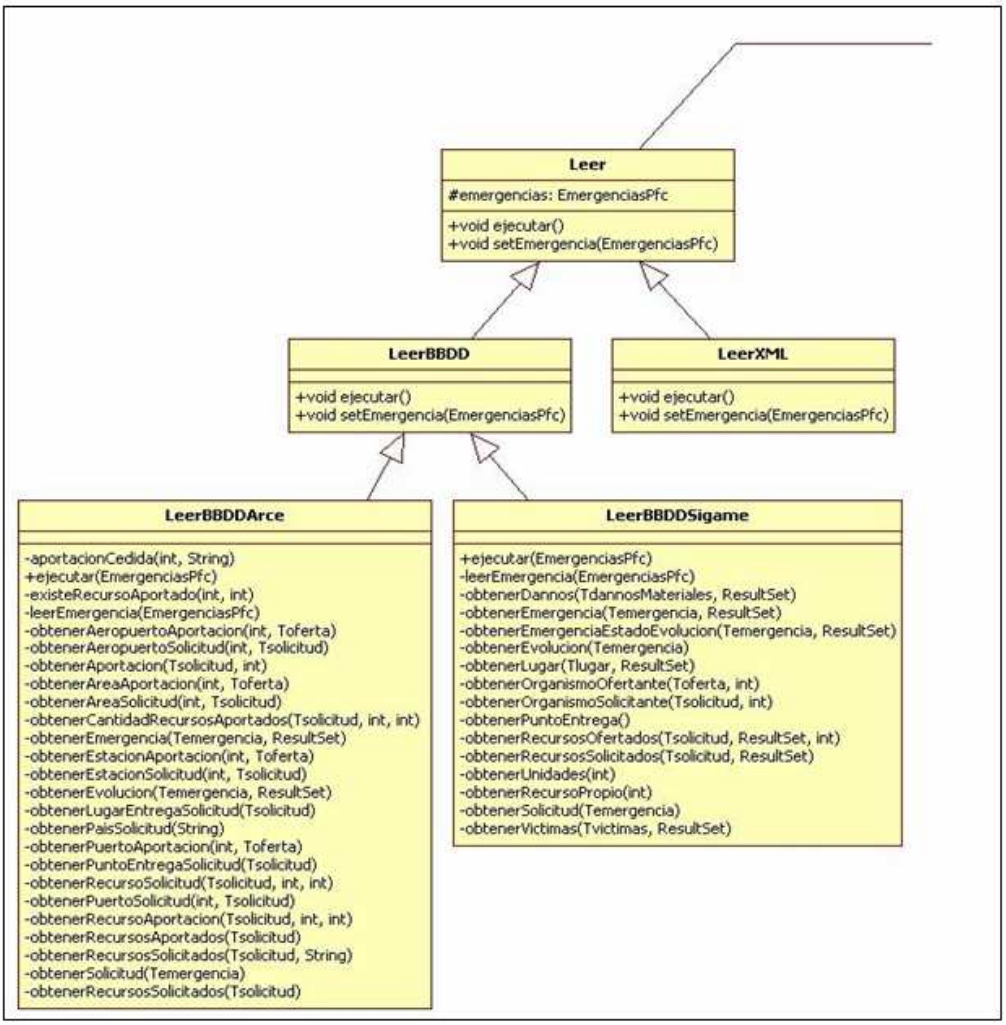


Ilustración 19 Módulo Lectura. Diagrama de clases.

La segunda imagen trata el módulo de escritura, tanto en ARCE como en SIGAME:

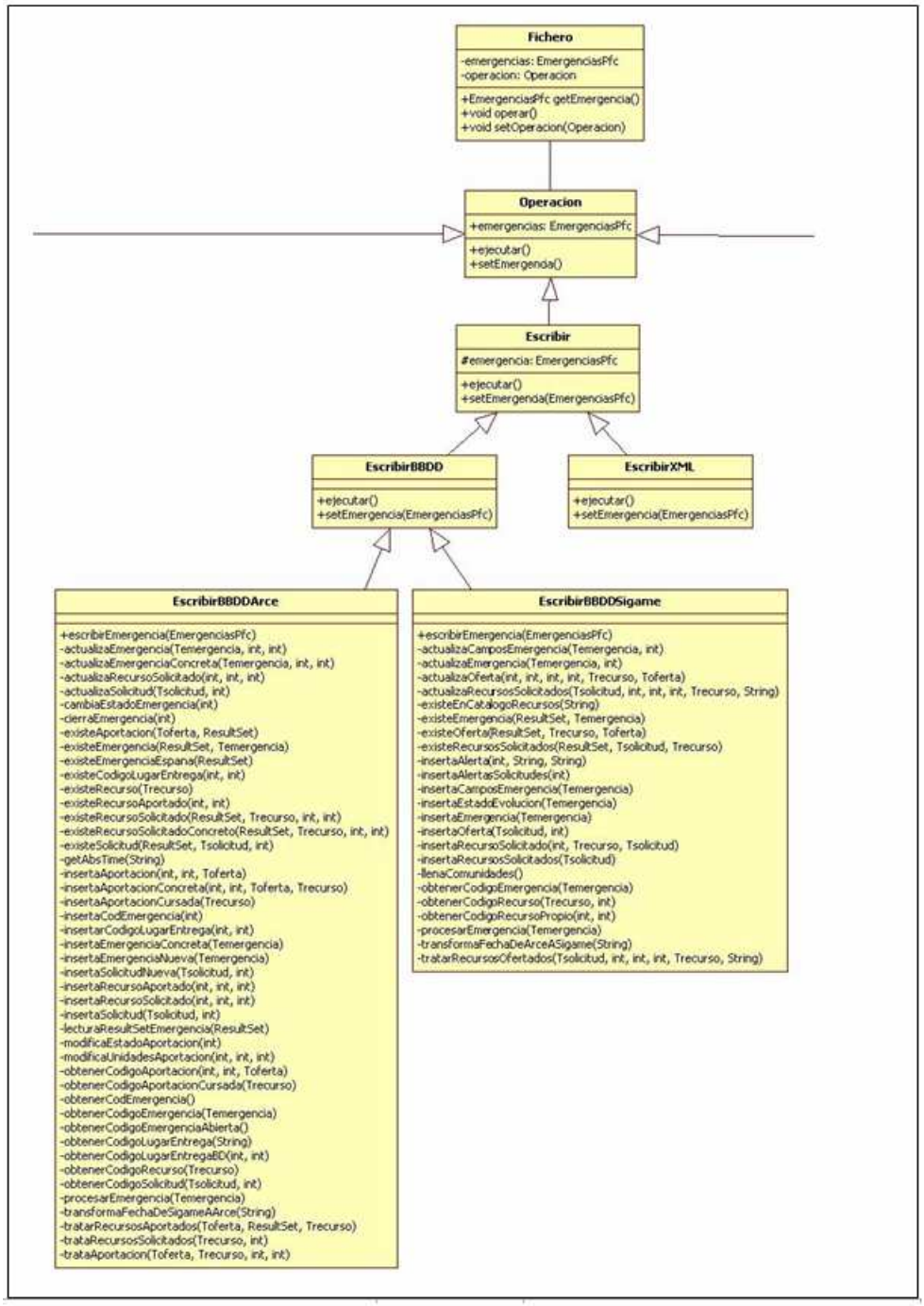


Ilustración 20 Módulo Escritura. Diagrama de clases.

La tercera imagen muestra el módulo de envío de ficheros por FTP:

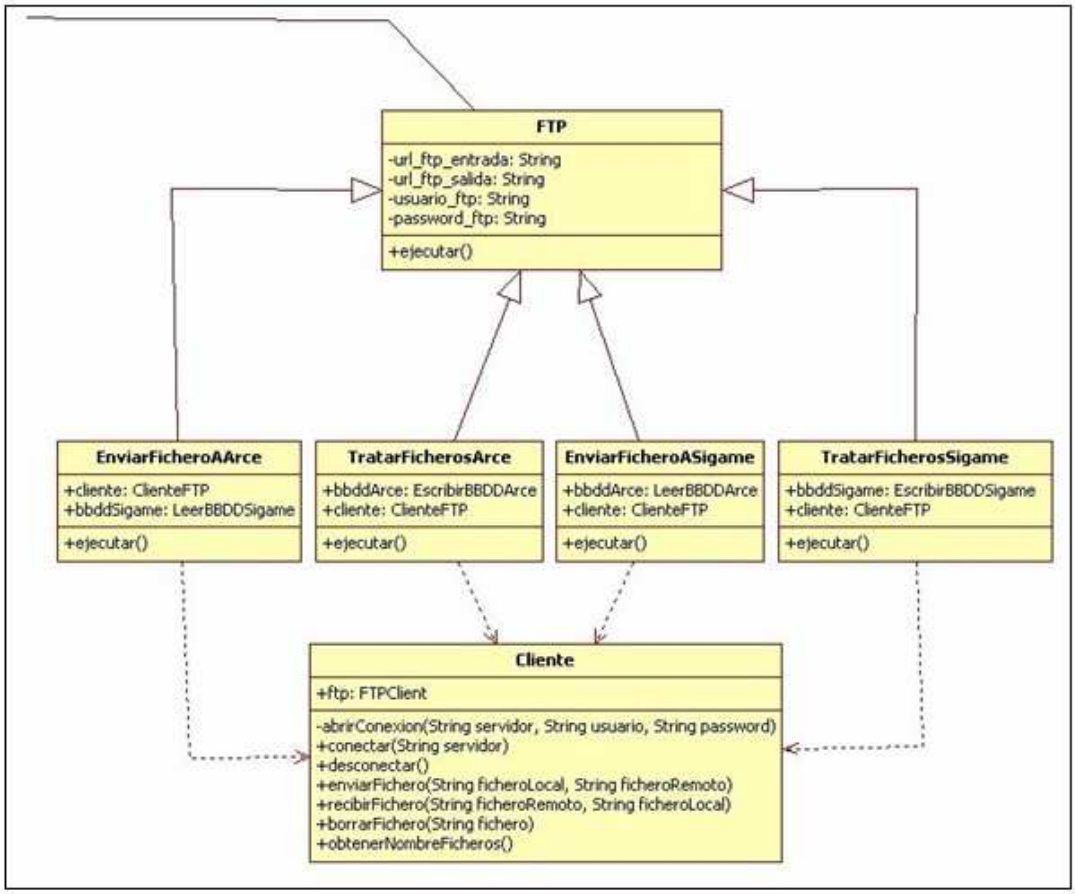


Ilustración 21 Módulo FTP. Diagrama de clases

Para explicar este apartado nos guiaremos por lo implementado en el código fuente realizada, la estructura que se implementará para el tratamiento del fichero contendrá varios paquetes, en primer lugar tendremos el paquete "generatedEmergenciasPFC" que almacenará las clases generadas por JAXB para el tratamiento de datos del fichero XML que generemos. El siguiente paquete que tendremos será el paquete "utilidades" con el cual haremos uso de clases tan conocidas como *BaseDeDatos* o *FicheroXML,* para tratar las conexiones de la base de datos o la creación del fichero entre otros métodos.

En el paquete "fichero" tendremos las clases que implementarán los métodos necesarios para la migración de datos concretamente, estarán diferenciados en sub-paquetes *ARCE* y *SIGAME*;

Cada uno de ellos tendrá dos clases LeerBBDD y EscribirBBDD que realizarán la lectura desde la base de datos para escribir la información en el fichero XML y la inserción en la base de datos correspondiente de los datos obtenidos desde el documento XML. Por último, tendremos el paquete "ftp" que contendrá las clases precisas para el envío del fichero desde un servidor a otro, tendremos una clase *cliente* que se conectará al servidor y enviará el fichero. Esto lo veremos en profundidad en el siguiente apartado.

Envío y recepción del XML

En este apartado se describirá el diseño de las clases que nos permitirán enviar y recibir los ficheros mediante FTP. FTP es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo. El Servicio FTP es ofrecido por la capa de Aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21.

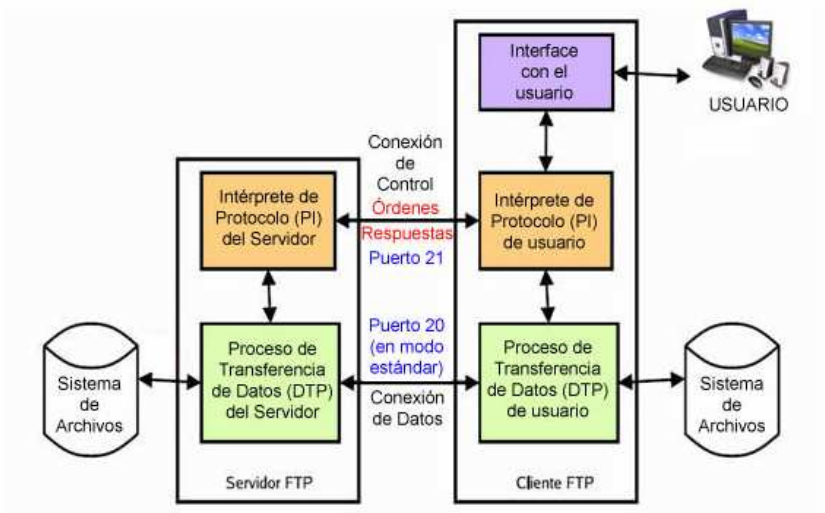


Ilustración 22 Diagrama de un servicio FTP

En la imagen anterior se describe el modelo que representa el diagrama de un servicio FTP, en el modelo, el intérprete de protocolo (PI) de usuario, inicia la conexión de control en el puerto 21. Las órdenes FTP estándar las genera el PI de usuario y se transmiten al proceso servidor a través

de la conexión de control. Las respuestas estándar se envían desde el PI del servidor al PI de usuario por la conexión de control como respuesta a las órdenes.

Estas órdenes FTP especifican parámetros para la conexión de datos (puerto de datos, modo de transferencia, tipo de representación y estructura) y la naturaleza de la operación sobre el sistema de archivos (almacenar, recuperar, añadir, borrar, etc.). El proceso de transferencia de datos (DTP) de usuario u otro proceso en su lugar, debe esperar a que el servidor inicie la conexión al puerto de datos especificado (puerto 20 en modo activo o estándar) y transferir los datos en función de los parámetros que se hayan especificado.

Vemos también en el diagrama que la comunicación entre cliente y servidor es independiente del sistema de archivos utilizado en cada ordenador, de manera que no importa que sus sistemas operativos sean distintos, porque las entidades que se comunican entre sí son los PI y los DTP, que usan el mismo protocolo estandarizado: el FTP.

También hay que destacar que la conexión de datos es bidireccional, es decir, se puede usar simultáneamente para enviar y para recibir, y no tiene por qué existir todo el tiempo que dura la conexión FTP.

En nuestro proyecto tenemos una clase llamada “ClienteFTP” que se encargarán de realizar los envíos y recepciones del fichero XML. Dependiendo del momento ARCE y SIGAME, serán cliente o serán servidor. El que sea servidor estará escuchando a la espera de una petición por parte del cliente, el cliente mandará un comando al servidor por el canal de control indicándole un número de puerto (aleatorio mayor de 1024), de manera que el servidor pueda abrirle una conexión de datos por donde transferirán los archivos y listados en el puerto especificado.

4.3 El intercambio de información

En este apartado se realiza un estudio en profundidad de la solución, está dividido en 3 grandes bloques, en el primer bloque se realizará el Análisis comenzando por el análisis de requisitos, que dividiremos en funcionales y no funcionales, a continuación podremos ver los casos de uso y diagrama de actividad de cada uno de ellos, seguidamente veremos los diagramas de secuencia. El segundo bloque será el diseño, donde podremos ver el diseño de alto nivel sobre

la arquitectura del sistema. Por último, se analiza el despliegue, donde se explica qué debemos hacer para utilizar nuestra solución.

Análisis

Este bloque contendrá la etapa de análisis del proyecto, es un proceso de refinamiento, modelado y especificación. Se refina en detalle el ámbito, y se crean modelos de los requisitos, flujo de información y control, y del comportamiento operativo. El análisis de requisitos nos permitirá especificar la función y rendimiento del proyecto, a continuación veremos el modelado de estos requisitos, serán los casos de uso del proyecto acompañados de los diagramas de actividad que nos ayudarán a comprender mejor el tratamiento funcional y comportamiento de la información.

Análisis de Requisitos

En este apartado se hace una especificación de requisitos del proyecto, tanto los requisitos funcionales como los no funcionales, a continuación, se muestran los diagramas necesarios para explicar la solución propuesta.

A continuación, podemos ver los requisitos del proyecto para llevar a cabo la comunicación entre los sistemas, esta interoperabilidad fuerza a hacer que los datos se transformen unos en otros para ello debemos de tener en cuenta también el tipo de datos, el usuario con el cual se acceda a la aplicación, etc.…

Los requisitos [13] se dividirán en dos tipos, los requisitos funcionales y los requisitos no funcionales.

Los primeros, es decir, los **requisitos funcionales** se encargan directamente del funcionamiento del proyecto, los procesos que se deben realizar y la información que debe contener. Los requisitos funcionales son los que describen el análisis, porque son las funcionalidades que se deben cumplir, y las acciones definidas en los casos de uso.

Los **requisitos no funcionales**, por su parte, son los que representan las propiedades del sistema que tienen que ver con el rendimiento, la seguridad…Se emplearán principalmente en la fase de diseño, se conocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar software.

Los requisitos no funcionales a su vez se dividen en cuatro tipos:

- Requisitos operativos: Son los requisitos que se refieran a la operatividad del sistema, es decir, vendrán dados por las necesidades del traspaso de información.
- Requisitos de rendimiento: Se refieren a los requisitos que tratan de especificar las características del rendimiento del sistema, como puede ser la velocidad.
- Requisitos de seguridad: Se referirán a los requisitos que tengan que ver con la seguridad informática, por ejemplo, a la autenticación de los usuarios de ARCE y SIGAME.
- Requisitos culturales y políticos: Factores legales que afectan al sistema.

Para nombrar a los requisitos emplearemos una nomenclatura, los requisitos se distinguirán por un tipo, que diferenciará si son funcionales y no funcionales, siendo “F” si el requisito es funcional y “NF” si por el contrario es no funcional, y un identificador para poder facilitar su trazabilidad a lo largo del desarrollo.

1. Requisitos Funcionales

Los requisitos funcionales que se deben cumplir se describen a continuación:

- RF01: Se debe asegurar la comunicación entre los distintos sistemas de emergencia.
- RF02: No debe de aparecer información duplicada en ninguna de las aplicaciones.
- RF03: Las bases de datos deben estar actualizadas (este requisito se supone puesto que son las aplicaciones las que mantienen actualizadas las bases de datos).
- RF04: No debe existir pérdida de información en el traspaso.
- RF05: El intercambio de datos sólo afectará a las emergencias. Esto quiere decir que en el caso de ARCE, no se actualizarán las noticias ni los comunicados de prensa. En ARCE, existe una primera aproximación a la emergencia que tampoco se debe actualizar con este proyecto, son la primera solicitud y al listado de las primeras aportaciones.

- RF06: Si SIGAME envía 2 emergencias a ARCE o en ARCE ya existe una emergencia con estado abierto de España, la información se actualizará si es la misma emergencia o se sobrescribirá la información con los datos nuevos.
- RF07: Los lugares de entrega deben introducirse en SIGAME en aquellas emergencias que la DGPCE considere conveniente traspasar a ARCE, estos lugares de entrega deben seguir un patrón.

2. Requisitos No Funcionales

Seguidamente, se muestran los requisitos no funcionales, divididos en sus cuatro tipos:

- Operativos:
  - RNF01: El programa se lanzará sin importar si la ejecución comienza en SIGAME o en ARCE.
  - RNF02: Para realizar la captura de datos se emplearán documentos XML.
  - RNF03: Para la codificación del código fuente el lenguaje de programación será Java.
- Seguridad:
  - RNF04: Los usuarios que utilicen las aplicaciones estarán autenticadas, es decir, las aplicaciones ya exigen un usuario y una contraseña para insertar, modificar y eliminar emergencias, aportaciones…, nuestro programa no enviará ninguna contraseña en el documento XML. El programa supone que las validaciones se realizan a nivel de aplicación.
  - RNF05: El sistema debe asegurar la integridad de los datos relacionados con las emergencias, tanto entrantes, como salientes, así como la procedencia de los mismos.
- Culturales y políticos:
  - RNF06: Ley de protección de datos de carácter personal

Casos de Uso y Diagramas de Actividad

Una vez conocemos los casos de uso de los sistemas de emergencia implicados en el proyecto tal y como vimos en el apartado 4.1 Revisión de los sistemas, vamos a explicar los casos de uso de nuestro proyecto. Nuestro proyecto tiene el caso de uso "Migrar información" que lo que hace es trasladar los datos de las emergencias activas de un sistema a otro, pero no lo mostramos como caso de uso en sí, puesto que realmente es mas importante explotarle y llegar a "Traspaso de información del sistema SIGAME a ARCE" y el caso de uso que explica la acción inversa, "Traslado de información del Sistema de Emergencias ARCE a SIGAME".

1. Casos de Uso

Un diagrama de casos de uso (*Use Case Diagram*) es una representación gráfica de parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones. Todo sistema tiene como mínimo un diagrama *Main Use Case*, que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso).

Un diagrama de casos de uso muestra, por tanto, los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

Main Use Case

En este diagrama se muestra los dos casos de uso generales del proyecto.

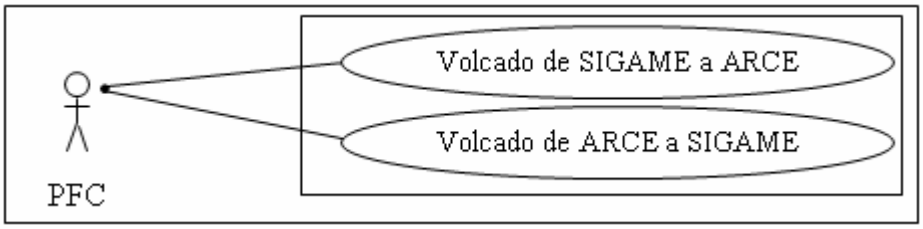


Ilustración 23 Casos de uso del proyecto

Explotación caso de uso "Volcado de Datos de SIGAME a ARCE":

En el siguiente diagrama podemos ver los casos de uso resultantes de la explotación del caso de uso “Volcado de Datos de SIGAME a ARCE”.

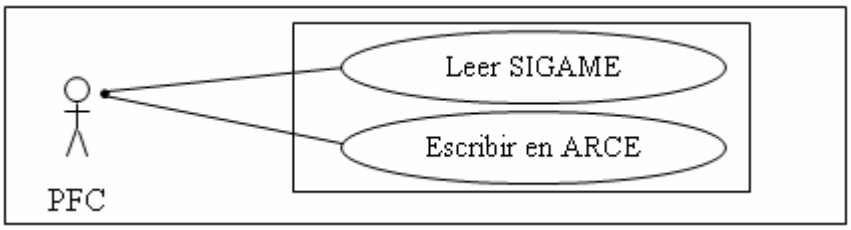


Ilustración 24 Caso de uso "Volcado de datos de SIGAME a ARCE"

Explotación caso de uso Leer de SIGAME:

Leer de SIGAME esta compuesto de los siguientes casos de uso:

- “Leer de Bd SIGAME” se encarga de realizar la extracción los datos necesarios de la Base de datos de SIGAME.
- “Escribir en XML” introduce los datos en el documento XML.

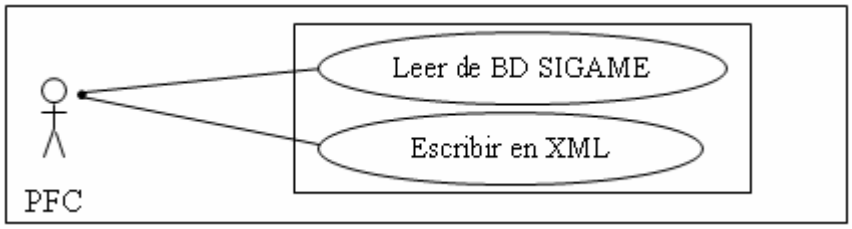


Ilustración 25 Caso de uso “Leer de SIGAME ”

Explotación caso de uso Escribir en ARCE:

Escribir en ARCE esta compuesto de los siguientes casos de uso:

- “Leer de XML” se encarga de realizar la extracción los datos necesarios del fichero XML.
- “Escribir en ARCE” introduce los datos leídos anteriormente del XML a la base de datos de ARCE.

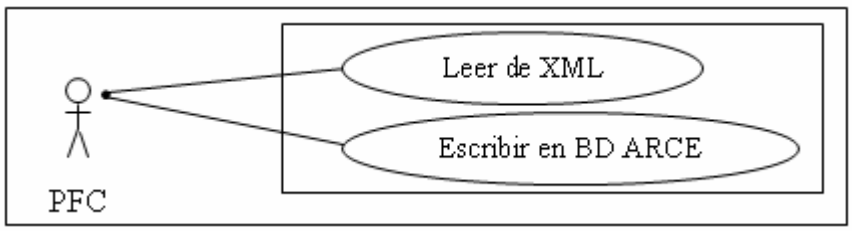


Ilustración 26 Caso de uso "Escribir en ARCE"

Explotación caso de uso Volcado de datos de ARCE a SIGAME:

En el siguiente diagrama podemos ver los casos de uso resultantes de la explotación del caso de uso "Volcado de datos de ARCE a SIGAME".

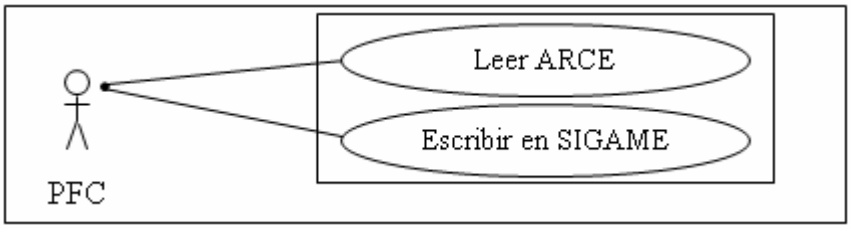


Ilustración 27 Caso de uso "Volcado de Datos de ARCE a SIGAME"

Explotación caso de uso Leer de ARCE:

Leer de ARCE está compuesto de los siguientes casos de uso:

- "Leer de Bd ARCE" se encarga de realizar la extracción los datos necesarios de la Base de datos de ARCE.
- "Escribir en XML" introduce los datos leídos en el caso de uso anterior en el XML.

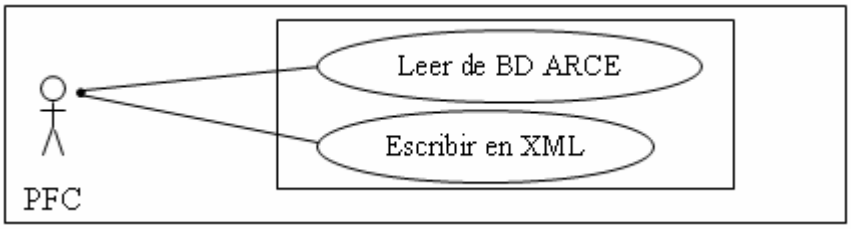


Ilustración 28 Caso de uso "Leer de ARCE"

Explotación caso de uso Escribir en SIGAME:

Escribir en SIGAME esta compuesto de los siguientes casos de uso:

- "Leer de XML" se encarga de realizar la extracción los datos necesarios del fichero XML.
- "Escribir en SIGAME" grabar los datos anteriormente leídos del XML en la base de datos de SIGAME.

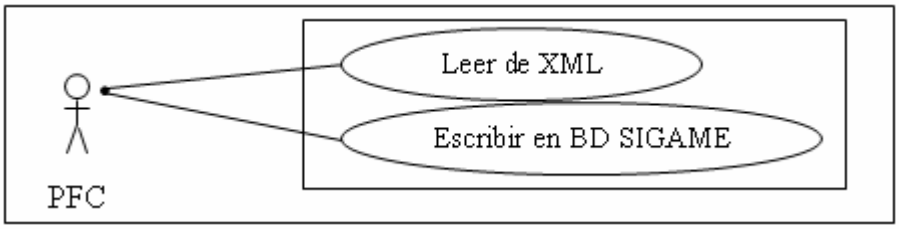


Ilustración 29 Caso de uso "Escribir en SIGAME"

A continuación, vemos la especificación de los casos de uso mencionados anteriormente, se mostrarán tablas que contendrán el nombre del caso de uso, su autor (quien lleva a cabo el caso de uso), la fecha en la cual se ejecuta, una descripción, los actores implicados, precondiciones, funcionamiento en estado normal, detalles del caso, flujo alternativo, y la postcondición (como será el estado del la aplicación una vez se lleve a cabo el caso de uso):

LEER DE SIGAME

Nombre: CONVERTIR LOS DATOS DESDE LA BASE DE DATOS SIGAME AL XML	
Autor:	PFC
Fecha:	--
Descripción: Permite la migración de datos desde la base de datos de SIGAME al documento XML correspondiente al sistema de emergencia SIGAME.	
Actores: Proceso automático.	
Precondiciones: 1. El campo "comunidad" de la tabla Solicitud de la base de datos de SIGAME debe ser "DGPCE" (Dirección General de Protección Civil). 2. El campo "tipo" de la tabla oferta determina si la entrega de los recursos se realizará en el origen o en el destino, dependiendo de si es origen o destino buscaremos la información en una tabla o en otra. 3. Se deben informar los campos "obs" de la tabla oferta si el tipo de dicha tabla es origen o el campo "observaciones" de la tabla recursos_solicitados si el campo tipo de la tabla Oferta es destino, para rellenar este campo se debe seguir un patrón especificado. 4. Las solicitudes que realizarán el traspaso serán las que tengan el estado "Abierto".	
Flujo normal: Transmitir los datos de la tabla solicitud. 1. Trasladar los datos de la tabla estado _ evolución. 2. Transferir los datos de la tabla recursos_solicitados. 3. Pasamos los datos de la tabla Oferta.	
Detalles: Las claves ajenas de las tablas estado _ evolución, recursos_solicitados y oferta deben concordar entre sí, siempre se comprobará si la clave ajena que referencia a la solicitud existe en la etiqueta Solicitud del Xml.	
Flujo alternativo:--	
Poscondición: Los datos han sido almacenados en el documento XML.	

Tabla 20 Especificación del Caso de Uso Leer de SIGAME 1

ESCRIBIR EN ARCE

Nombre: CONVERTIR LOS DATOS DEL XML A LA BASE DE DATOS DE ARCE	
Autor:	PFC
Fecha:	--
Descripción: Permite la migración de datos desde el documento Xml correspondiente a ARCE a la base de datos de ARCE.	
Actores: Proceso automático.	
Precondiciones: En ARCE solo se permite una emergencia por país, de modo que España solo podrá abrir una emergencia en ARCE, en caso de existir una se actualizará si no existiera se daría de alta.	
Flujo normal: 1. Transmitir los datos de la Emergencia. 2. Trasladar los datos de la Solicitud 3. Transferir los datos de los recursos_solicitados 4. Pasamos los datos de los lugares 5. Pasamos los datos de los lugares_entrega 6. Transmitir los datos de la aportación.	
Detalles: Todos los datos que hagan referencia a tablas anteriores deben existir. Los datos con los que informaremos los lugares de entrega deben existir y seguir el patrón	
Flujo alternativo: --	
Poscondición: Los datos han sido almacenados en la bases de Datos de ARCE.	

Tabla 21 Especificación de Caso del Uso Grabar en ARCE 2

En este apartado se da la particularidad de que los lugares de entrega no existen directamente

en SIGAME, y en este caso para ARCE son obligatorios, por ello en SIGAME se obliga a seguir un patrón a la hora de insertar las observaciones de la solicitud y de la oferta, que no es más que poner el literal, "LUGAR DE ENTREGA: ", y a continuación, por ejemplo, "Aeropuerto de Madrid", con lo cual identificamos que el lugar de entrega es un aeropuerto, en ARCE, la entidad aportación necesita un código para el lugar de entrega de los medios, pero los lugares no son los mismos en SIGAME y en ARCE, para ello, se ha creado en ARCE una entrada para cada tabla que referencia al lugar (aeropuerto, área, estación y puerto), en primer lugar se inserta cuatro filas en la tabla "cod_lugar" para poder asociarlas desde la oferta,estos códigos que insertamos deben ser un código que no exista en la tabla, se ha decidido que serán 996 para la tabla "aeropuerto", 997 para la tabla "area", 998 para la tabla "estacion" y 999 para la tabla "puerto", también le pondremos un nombre pero no tiene que ser siempre el nombre correcto del aeropuerto,…, así pues le pondremos siempre como nombre "Sigame", y al ser una emergencia de España siempre que se migre la información desde SIGAME, el código del país que ARCE tiene asignado a España es 11, los datos que se insertan son:

Tabla	Código	Nombre	País
Aeropuerto	996	Sigame	11
Area	997	Sigame	11
Estacion	998	Sigame	11
Puerto	999	Sigame	11

Tabla 22 Lugares de entrega

*Siendo 11 el código perteneciente a España en ARCE.

Cuando se inserta el lugar de entrega en ARCE debe haber un campo observaciones que diga donde se entregarán los medios, en caso contrario se buscará una forma de informar de este hecho, por ejemplo una llamada de teléfono. En SIGAME solo necesitaremos un lugar de entrega que coincida con el origen o destino de la oferta, y los datos del lugar se encuentran en las descripciones.

Ahora mostramos los casos de uso que describen el proceso inverso, es decir, leeremos del sistema ARCE e insertaremos los datos en el documento XML y estos seguidamente se almacenarán sistema SIGAME.

LEER DE ARCE

Nombre: CONVERTIR LOS DATOS DESDE LA BASE DE DATOS DE ARCE AL XML	
Autor:	PFC
Fecha:	--
Descripción: Permite la migración de datos desde la base de datos de ARCE al documento XML correspondiente a ARCE.	
Actores: Proceso automático.	
Precondiciones: 1. Consultaremos las tablas y tomamos las emergencias cuyo usuario tenga una entrada en la tabla Roles cuyo rol sea “1”, “41” o “42”. 2. El estado de la emergencia debe ser “A”, es decir, abierta.	
Flujo normal: 1. Transmitir los datos de la tabla emergencia. 2. Trasladar los datos de la tabla solicitud. 3. Transferir los datos de la tabla lugar_entrega. 4. Pasamos los datos de la tabla lugar. 5. Transmitir los datos de la tabla recursos_solicitados. 6. Transferir los datos de la tabla aportaciones. 7. Pasamos los datos de la tabla recursos_aportados.	
Detalles: Las claves ajenas de las tablas deben concordar entre sí, siempre se comprobará si la clave ajena que referencia a la solicitud existe en la etiqueta Solicitud del XML.	
Flujo alternativo:--	
Poscondición: Los datos han sido almacenados en el documento XML correspondiente a ARCE.	

Tabla 23 Especificación de Caso del Uso Leer de ARCE 1

GRABAR EN SIGAME

Nombre: CONVERTIR LOS DATOS DEL XML A LA BASE DE DATOS DE SIGAME	
Autor:	PFC
Fecha:	--
Descripción: Permite la migración de datos desde el documento XML correspondiente a SIGAME a la base de datos de SIGAME.	
Actores: Proceso automático.	
Precondiciones: El usuario que tomará como propias las emergencias procedentes de ARCE será DGPCE, puesto que el país no estará definido en SIGAME.	
Flujo normal: 1. Transmitir los datos a la tabla solicitud 2. Trasladar los datos a la tabla estado _ evolución 3. Transferir los datos a la tabla recursos_solicitados 4. Pasamos los datos a la tabla oferta.	
Detalles: Todos los datos que hagan referencia a tablas anteriores deben existir.	
Flujo alternativo: --	
Poscondición: Los datos han sido almacenados en la base de datos de SIGAME.	

Tabla 24 Especificación de Caso del Uso Grabar en SIGAME 2

Un diagrama de Actividad demuestra la serie de actividades que deben ser realizadas en un caso de uso, así como las distintas rutas que pueden irse desencadenando en el caso de uso

Un diagrama de actividad es utilizado en conjunción de un diagrama de casos de uso para auxiliar a los miembros del equipo de desarrollo a entender como es utilizado el sistema y como

reacciona en deteminados eventos. Se pudiera considerar que un diagrama de actividad describe el *problema*.

En los siguientes diagramas nos encontraremos con los siguientes elementos que componen un diagrama de Actividad.

Inicio: El inicio de un diagrama de actividad es representado por un círculo de color negro sólido.

Actividad: Una actividad representa la acción que será realizada por el sistema la cual es representada dentro de un ovalo.

Transición: Una transición ocurre cuando se lleva acabo el cambio de una actividad a otra, la transición es representada simplemente por una línea con una flecha en su terminación para indicar dirección.

Fin: El fin de un diagrama de actividad es representado por un círculo de color negro sólido y rodeándole uno de color blanco.

A continuación veremos los diagramas de actividad que apoyan a los casos de uso de nuestro proyecto.

LEER DE LA BASE DE DATOS SIGAME Y ESCRIBIR EN EL FICHERO XML

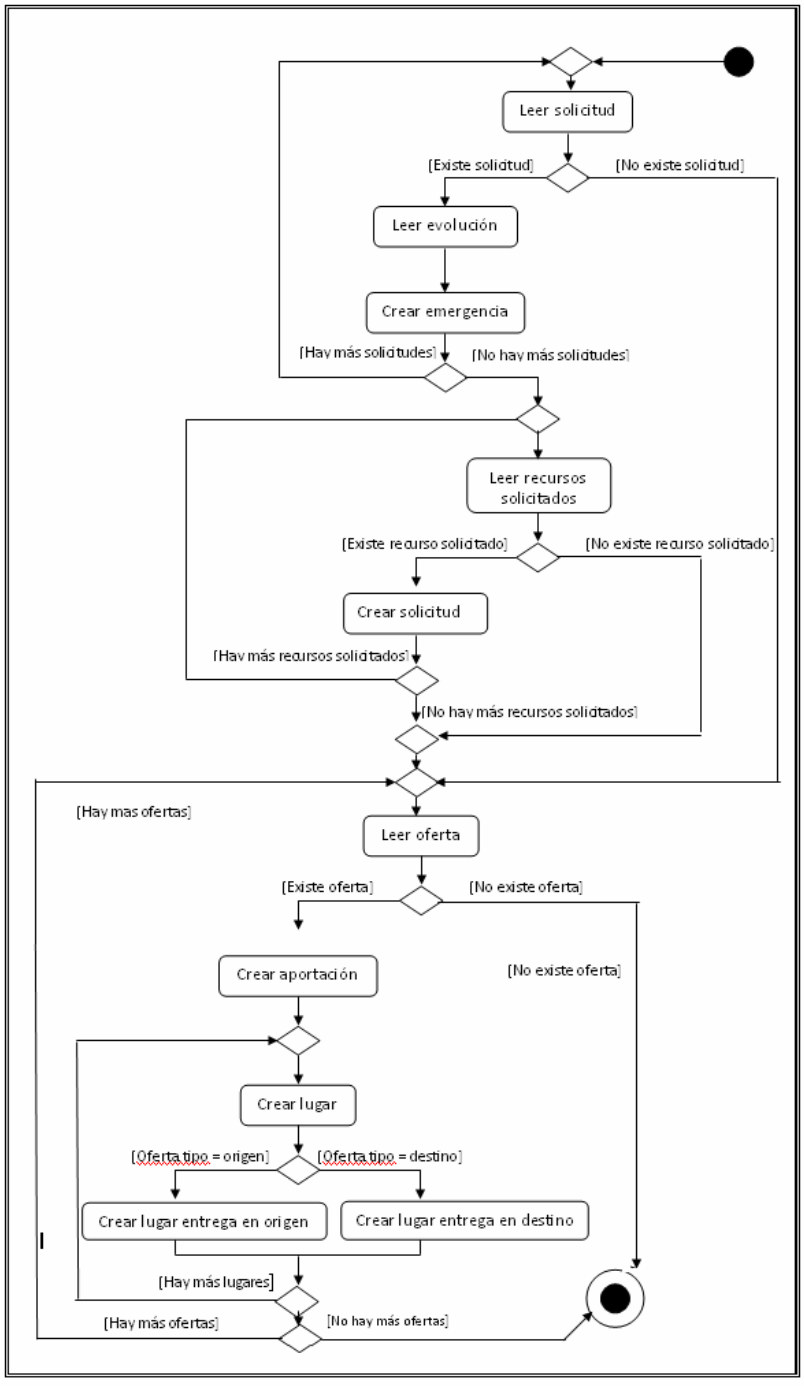


Ilustración 30 Leer de SIGAME y escribir en el XML

Se debe cumplir que el contenido del campo “comunidad” de la tabla “solicitud” sea “DGPCE” ya que la condición para traspasar la información de la emergencia desde SIGAME hasta ARCE, es que dicha emergencia haya sido autorizada por la Dirección General de Protección Civil, del mismo modo se debe comprobar que el estado de las emergencias que se migren será “Abierto”, puesto que no tendría sentido traspasar emergencias cerradas que ya no necesiten ayuda exterior.

En el diagrama anterior (Ilustración 30 Leer de SIGAME y escribir en el XML) podemos ver las actividades que se realizan para realizar el paso de la información desde la base de datos al fichero, en primer lugar se tratará la solicitud con la evolución de la misma para crear lo que será la emergencia, una vez tenemos la emergencia, se asociarán a ella los recursos solicitados. Y por último, las aportaciones con los valores obtenidos de la oferta de SIGAME crearemos la aportación, y los recursos aportados, en este punto obtendremos también los lugares donde se hará la entrega de los medios aportados.

LEER DEL FICHERO XML Y ESCRIBIR EN LA BASE DE DATOS ARCE

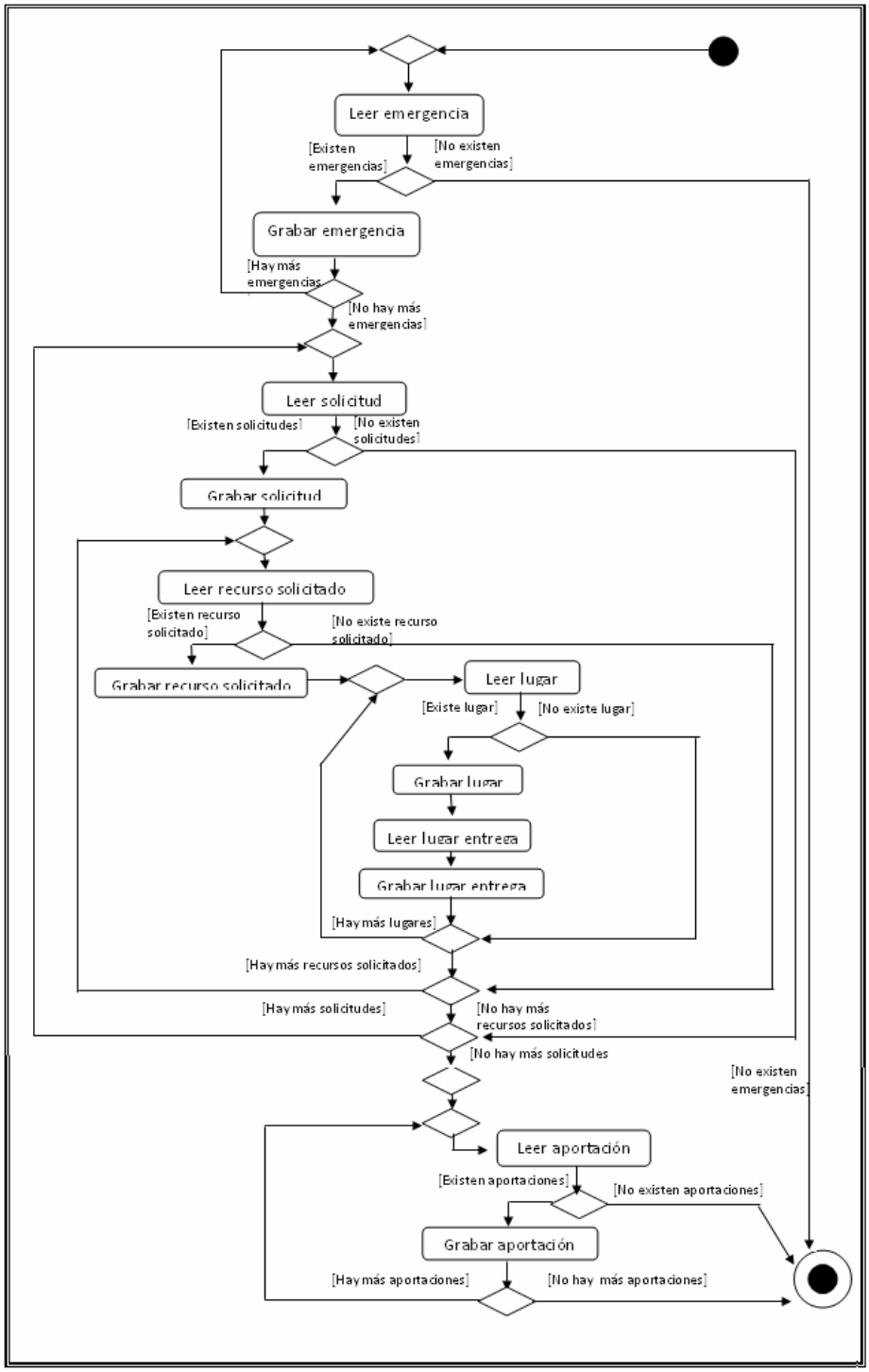


Ilustración 31 Leer del fichero XML y escribir en ARCE

Se debe cumplir que el contenido que vamos a introducir en la base de datos del sistema ARCE sean datos reales, existentes en el documento XML. En el diagrama anterior (Ilustración 31 Leer del fichero XML y escribir en ARCE) podemos ver las actividades que se realizan para llevar a cabo el paso de la información al fichero XML a la base de datos correspondiente a ARCE, en primer lugar se lo que hemos guardado antes como “emergencia”, el bloque siguiente trata las solicitudes y los recursos solicitados, en este momento es cuando separa los recursos solicitados y los lugares de entrega. Por último, se trata las aportaciones, obteniéndose así al insertar en la base de datos, la modificación de las tablas correspondientes, es decir, al realizar esta inserción se debe actualizar e insertar al mismo tiempo en las tablas donde se almacenen los recursos aportados y las aportaciones cursadas.

LEER DE DE LA BASE DE DATOS ARCE Y ESCRIBIE EN EL FICHERO XML

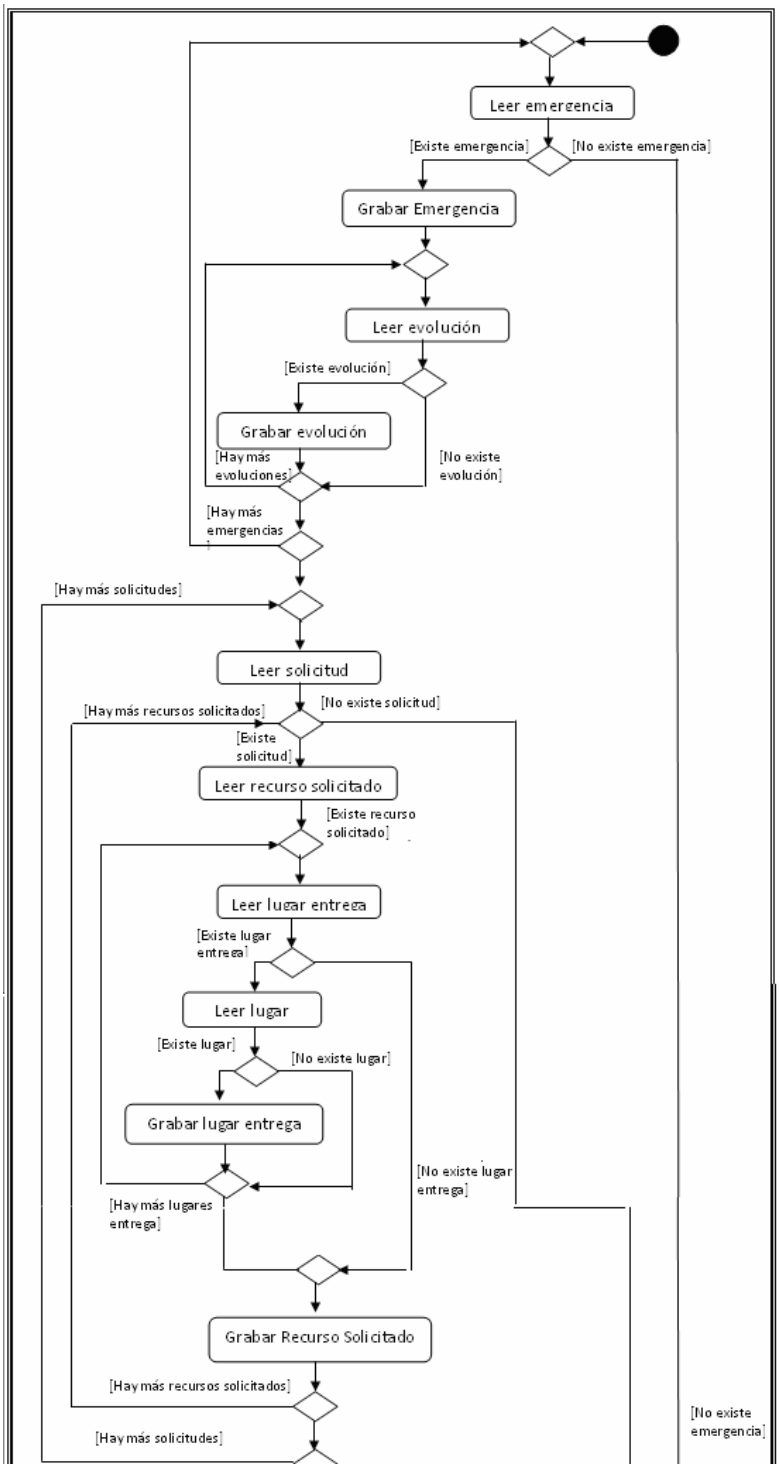


Ilustración 32 Leer de la base de datos ARCE y escribir en SIGAME

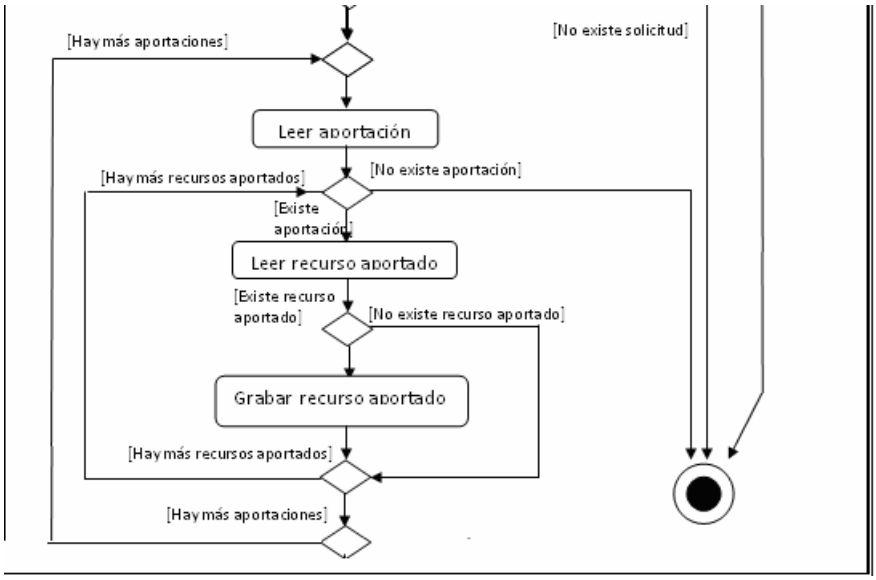


Ilustración 30 Leer de la base de datos ARCE y escribir en SIGAME (cont.)

En el caso del volcado de información desde la base de datos ARCE al documento XML, se tendrá en cuenta que el contenido del campo "rol" asociado al usuario que inserta la emergencia en el sistema debe ser "1", "41" o "42", que corresponderá a los usuarios a los cuales se les permite insertar en el sistema. Al igual que el volcado que realizamos en SIGAME el estado que posea la emergencia en cuestión, debe ser "abierto", ya que no tendría sentido pasar de un sistema a otro, emergencias que ya no estén en uso.

Los roles permitidos para dar de alta emergencias en ARCE son N4a, N4b y N4c, estos roles corresponden al Centro Coordinador Operativo H24 del Organismo asociado. Pueden relacionarse con todos los usuarios en situación cotidiana y en situación de emergencia.

En situación de emergencia, estos usuarios están habilitados (N4a y N4b) para crear la situación de emergencia y las actualizaciones. También serán los encargados de formular las peticiones de ayuda exterior y negociar las aportaciones de su país con un organismo asociado demandante. Por último, este rol (N4a y N4b) están autorizados para incorporar ARCE a usuarios N5, N6 y N7.

En el caso de establecerse un Puesto de Mando Avanzado se le asignará al responsable del

mismo un rol N4c. No se puede relacionar con otros niveles similares de otros países.

Comenzaremos tratando las emergencias y su evolución, para continuar con sus solicitudes, cada una de estas solicitudes tendrá que tratar sus recursos solicitados, en este caso al provenir la información de ARCE poseerán un lugar de entrega que estará asociado a cada solicitud de medios.

Para acabar, se leerán las aportaciones, y sus recursos aportados.

LEER DEL FICHERO XML Y ESCRIBIR EN LA BASE DE DATOS SIGAME

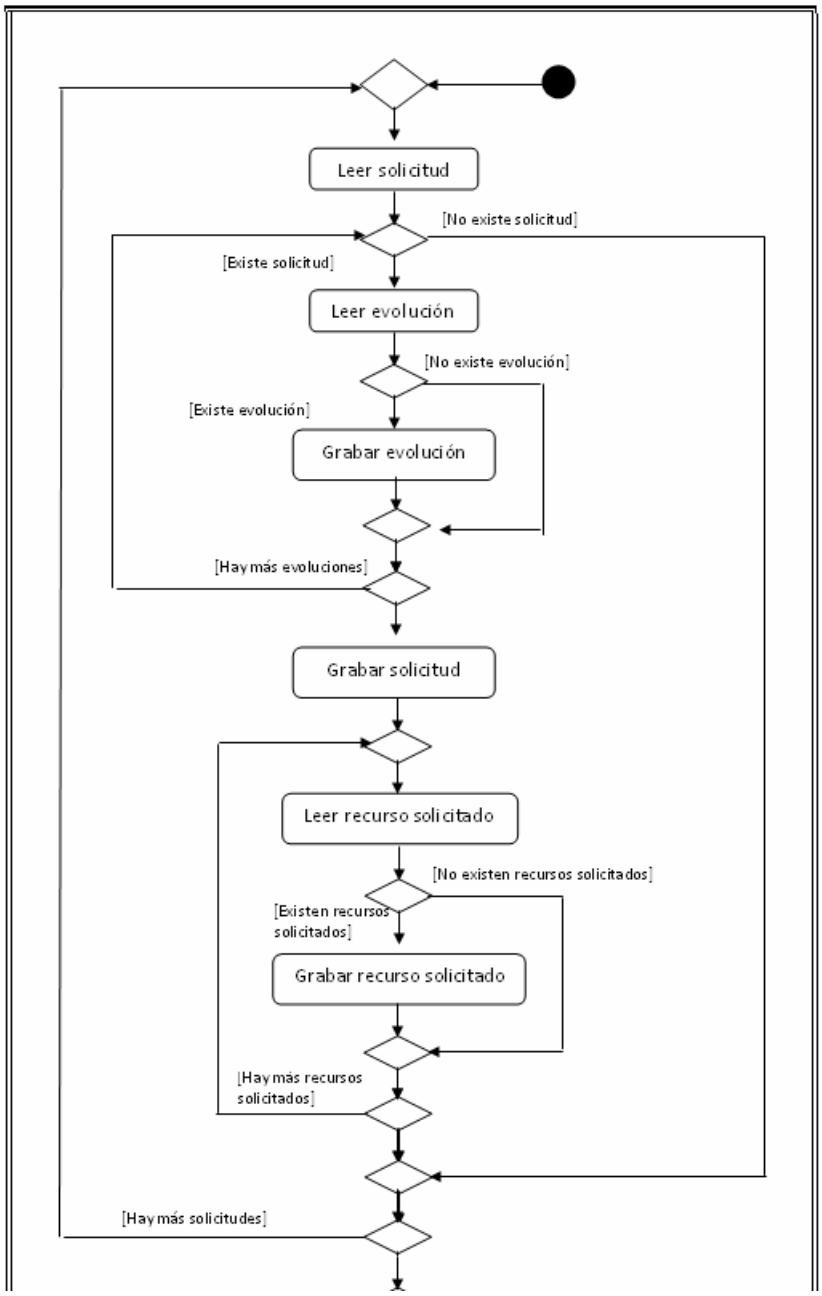


Ilustración 33 Leer del XML y escribir en la base de datos SIGAME

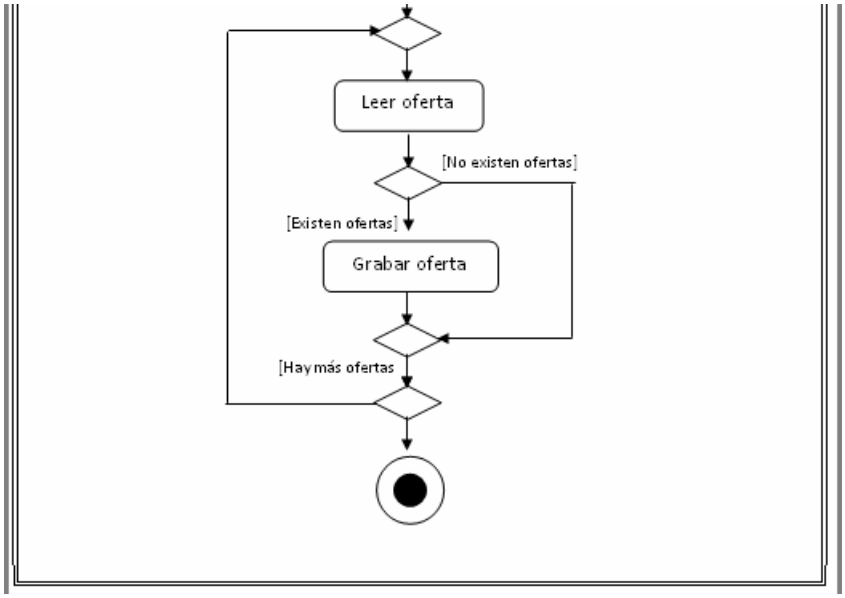


Ilustración 34 Leer del XML y escribir en la base de datos SIGAME (cont.)

Se debe cumplir que el contenido que vamos a introducir en la base de datos del sistema SIGAME sean datos reales, existentes en el documento XML.. En el diagrama anterior (Ilustración 33 Leer del XML y escribir en la base de datos SIGAME) podemos ver las actividades que se realizan para llevar a cabo el paso de la información al fichero XML a la base de datos correspondiente a SIGAME, en primer lugar se lo que hemos guardado antes como "solicitud" tratando al mismo tiempo sus evoluciones, el bloque siguiente trata los recursos solicitados correspondientes y para terminar se insertarán las ofertas. Teniendo en cuenta que la tabla que almacena la información de las ofertas contiene los recursos cedidos para cada una de ellas.

2. Diagramas de Secuencia

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. Seguidamente, veremos el diagrama de secuencia que representa los casos de uso anteriormente descritos.

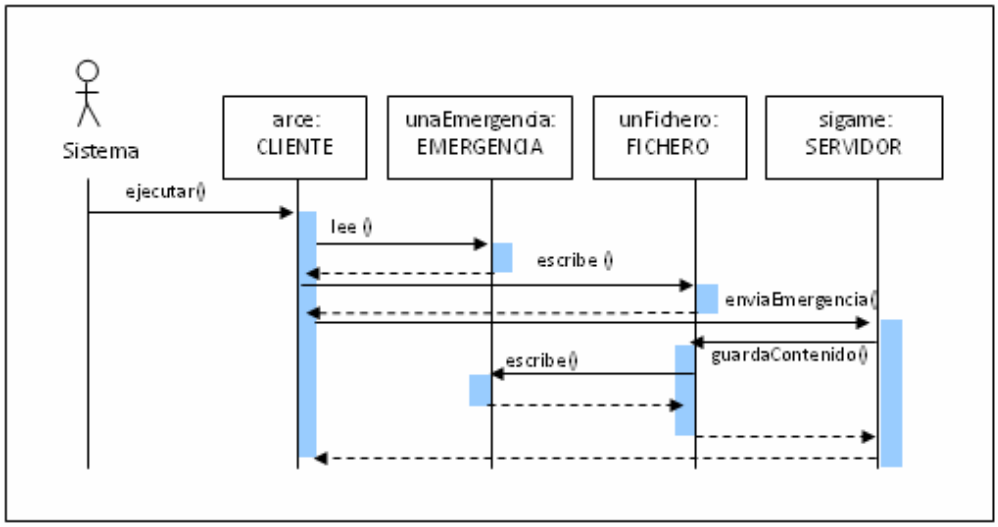


Ilustración 35 Diagrama de Secuencia del proyecto

En la imagen anterior, Ilustración 35 Diagrama de Secuencia del proyecto, se muestra el servicio que se va a llevar a cabo en el proyecto, será "Añadir emergencias", el orden de realización sería el siguiente, el cliente se conecta al servidor y le manda las últimas emergencias, otra opción sería "Recuperar emergencias" en la cual el cliente se conecta al servidor y le pide las últimas emergencias para actualizarlas en su sistema, elegimos realizar la opción en la cual el servicio es "Añadir emergencias" puesto que es una secuencia más natural y consume menos recursos y de esta manera podríamos ejecutar varios clientes en paralelo a la vez, para por ejemplo, en el caso de existir muchas emergencias y que el fichero generado se hiciera demasiado grande esto provocaría el incremento del tiempo de ejecución, para esto podríamos ejecutar varios clientes al mismo tiempo que llamaran al servidor pasándole ficheros más pequeños.

Por el momento, la opción elegida lleva a cabo la siguiente secuencia, en primer lugar el sistema lanzará un cron que ejecute el cliente, un servicio "CRON" (Command Scheduler) sirve para programar ciertas tareas que se realizarán en el futuro y de forma periódica. Este servicio es el que manda ejecutar el cliente e iniciará el proceso, el cliente recupera la información de una emergencia, y genera un fichero con esta información que hemos obtenido de la emergencia y le

envía este fichero al servidor, el servidor al recibir el fichero lo lee y genera la emergencia en el sistema.

Diseño

En este apartado se produce un modelo o representación de una entidad que se va a construir posteriormente. Una vez hemos analizado el problema y definido las solucionas más adecuadas, debemos concretar la solución y describirla en su totalidad, y con el detalle necesario para su posterior transformación (construcción o implementación) en objeto. En este apartado se mostrarán dos secciones, en primer lugar veremos el diseño de la arquitectura del sistema, y a continuación, el diseño detallado, donde veremos la representación de la información, el tratamiento de los XML, y el envío y la recepción de dicho XML.

Diseño de alto nivel sobre la arquitectura del sistema

El proyecto presenta una arquitectura cliente/servidor [14], esto significa que estamos frente a la plataforma abierta por excelencia. Ciertamente, las posibilidades de igualar o nivelar distintos productos o aplicaciones de distintos proveedores nos brinda la oportunidad de hacer una gran variedad de combinaciones de clientes y servidores.

La tecnología cliente/servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. Desde el punto de vista funcional, se puede definir la computación cliente/servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aun en entornos multiplataforma.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor, y este envía uno o varios mensajes con la respuesta. En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras. Además como veremos en el modelo de implementación, el concepto es utilizado en forma constante para varias funciones e implementado de distintas formas.

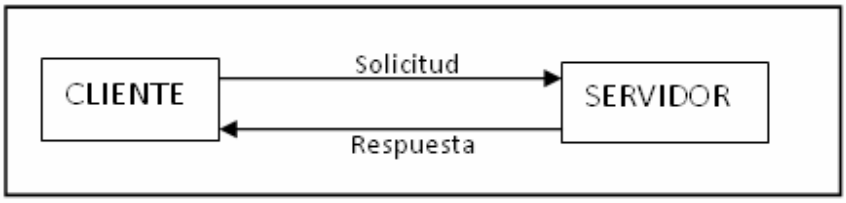


Ilustración 36 Modelo cliente-servidor

En nuestro caso, tendremos dos máquinas y en las dos tendremos un cliente y un servidor que se intercambian información.

La arquitectura cliente/servidor es un modelo basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Además esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos. De estas líneas se desprenden los tres elementos fundamentales sobre los cuales se desarrollan e implantan los sistemas cliente/servidor: el proceso cliente que es quien inicia el diálogo, el proceso servidor que pasivamente espera a que lleguen peticiones de servicio y el middleware que corresponde a la interfaz que provee la conectividad entre el cliente y el servidor para poder intercambiar mensajes.

En nuestro caso tenemos una estructura en la cual existirán dos máquinas las cuales contendrán cada una un cliente y un servidor, y se intercambiarán la información, a continuación podemos ver como es la arquitectura en nuestro sistema:

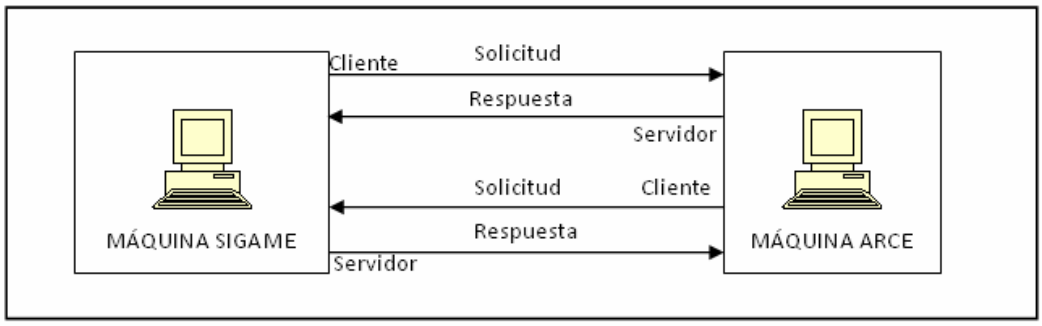


Ilustración 37 Modelo cliente- servidor del proyecto

En esta figura podemos ver como ARCE y SIGAME son al mismo tiempo clientes y servidores dependiendo del momento, cuando la información pase del sistema ARCE al sistema SIGAME, ARCE será el servidor y SIGAME será el cliente, al contrario sería si la información se pasa de SIGAME al sistema ARCE.

El sistema que sea cliente debe formular los requerimientos y pasarlos al servidor, se lo conoce con el término front-end. Además deberá realizar más funciones como, administrar la interfaz de usuario, procesar la lógica de la aplicación y hacer validaciones locales, generar requerimientos de bases de datos, recibir resultados del servidor y formatear resultados.

El sistema que sea servidor será el encargado de atender al cliente, SIGAME o ARCE dependiendo del caso. Al proceso servidor se lo conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos, deberá cumplir funciones como aceptar los requerimientos de bases de datos que hacen los clientes, procesar requerimientos de bases de datos, formatear datos para trasmitirlos a los clientes y procesar la lógica de la aplicación y realizar validaciones.

4.4 Despliegue

En este apartado se describirán los aspectos que intervienen en el despliegue de la aplicación, será un servicio entre sistemas de emergencia, como vimos en la Ilustración 36 Modelo cliente-servidor, el cliente solicita una petición al servidor y este le enviará la información correspondiente que espera.

Para el correcto funcionamiento del proyecto de fin de carrera debemos tener usuario en cada una de las máquinas, estas máquinas deben tener el servicio FTP habilitado. Tendremos dos ficheros de propiedades, que contendrían los datos de conexión entre otros, el fichero que guarda los datos de SIGAME (SigamePropiedades):

```
url_bbdd=jdbc:mysql://163.117.137.116:3306/Sigame_Paloma_pfc
usuario_bbdd=paloma_pfc
password_bbdd=paloma_pfc*pass
```

```
driver_bbdd=com.mysql.jdbc.Driver
url_ftp_entrada=docencia.dei.inf.uc3m.es
url_ftp_salida=arce.dei.inf.uc3m.es
puerto_ftp=21
usuario_ftp=paloma
password_ftp=paloma*pass
directorio_ftp_entrada=Paloma/PFC/src/es/uc3m/pfc/emergenciasPFC.xml
directorio_ftp_salida=emergenciasPFC.xml
```

El fichero donde se leerán los datos de conexiones a ARCE, llamado ArcePropiedades, contiene la siguiente información:

```
url_bbdd=jdbc:postgresql://arce.dei.inf.uc3m.es:5432/arce_paloma_pfc
usuario_bbdd=palomapfc
password_bbdd=postgres*paloma
driver_bbdd=org.postgresql.Driver
url_ftp_entrada=arce.dei.inf.uc3m.es
url_ftp_salida= Paloma/PFC/src/es/uc3m/pfc/emergenciasPFC.xml
puerto_ftp=21
usuario_ftp=paloma
password_ftp=paloma*pass
```

En la siguiente imagen podemos ver cómo funcionará el proyecto, si empleamos como ejemplo la migración de la información de SIGAME a ARCE, la DGPCE desde la aplicación u otro método pide que la emergencia que selecciona debe transferirse al sistema ARCE, en ese momento se lleva a cabo el método "*ejecutar ()*", que se encarga de leer de la base de datos de SIGAME y lo almacena en el fichero, una vez acaba, lo envía mediante FTP a ARCE. ARCE tiene un proceso continuamente preguntando si hay algún fichero nuevo en la carpeta determinada, donde SIGAME deja el archivo, cuando ARCE lo detecta lo procesa, es decir, inserta la información en su base de datos, y a continuación, lo borra.

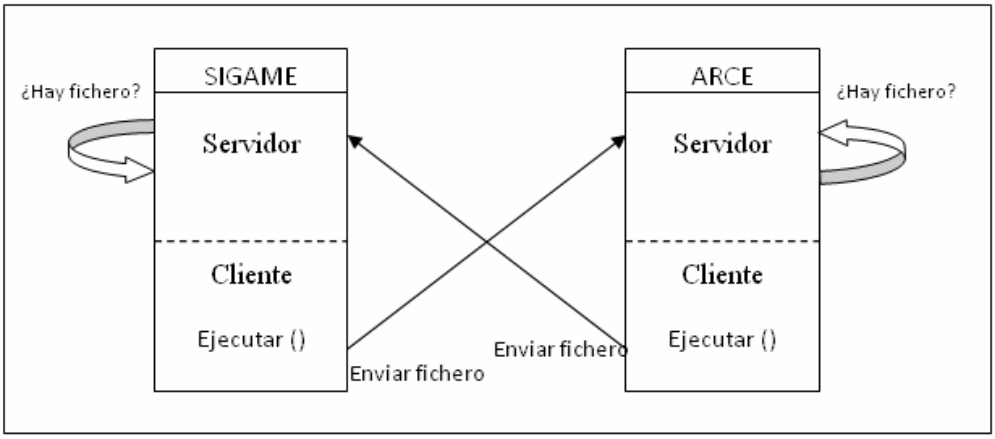


Ilustración 38 Funcionamiento del PFC

5 Evaluación

A lo largo de todo el proyecto hemos explicado en profundidad cada uno de los pasos que se llevarán a cabo a la hora de realizar el traspaso de emergencias, a continuación, podremos ver todo el proceso, descrito por la s pruebas.

El objetivo que se pretende conseguir con este apartado es demostrar que el traspaso se realiza correctamente y funciona tal y como se especifica en los requisitos. Para ello se definirán varios casos de prueba que comprenderán toda la funcionalidad descrita, a continuación se mostrarán los resultados obtenidos.

Desde un principio se ha diferenciado la transformación de dos maneras, la transferencia de datos desde el sistema de emergencia ARCE hacia el sistema de emergencia SIGAME y la transformación inversa. Realmente no debe importar quien de los dos sistemas comience el traspaso, sino que la información sea lo mas fiel posible a la realidad.

Se debe tener en cuenta que los casos de prueba que se llevarán a cabo contienen datos correctos, por ejemplo en el momento de insertar un lugar de entrega en SIGAME, debemos saber que esta información la requiere ARCE, y se tendrá que introducir la información con el formato especificado en el apartado 4.3 El intercambio de Información, en el momento en el que especificamos los Casos de Uso del Proyecto.

Una vez acabado el análisis, el diseño y el desarrollo debemos centrarnos en las pruebas, existen dos tipos de pruebas, las pruebas funcionales o de caja negra y las pruebas unitarias o de caja blanca, las primeras tienen por objetivo probar que el proyecto (en nuestro caso) cumple las funciones para lo que ha sido creado, con lo cual debería de cumplir la interoperabilidad entre ARCE y SIGAME, no se enfoca su atención a como se generan las respuestas del sistema, básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida. Las segundas, es decir, las pruebas unitarias, nos permiten determinar si un modulo del programa esta listo y correctamente terminado.

5.1 Proceso de evaluación

En la siguiente tabla se almacenan los datos de los casos de prueba que hemos obviado necesarios para demostrar el cumplimiento de los requisitos. Veremos únicamente el resumen de los casos de prueba, a continuación se mostrarán tablas con los casos de prueba [15] concretos por módulo, de esta manera comprobaremos el correcto funcionamiento del programa.

Identificador del Caso de Prueba	CP1	CP2
Módulo a probar	ARCE → SIGAME	SIGAME → ARCE
Descripción del caso	Envía la información de ARCE a SIGAME.	Envía la información de ARCE a SIGAME
Prerrequisitos	--	--
Resultado esperado	Información vista desde SIGAME.	Información vista desde ARCE.
Resultado obtenido	OK	OK
Estado	Concluida.	Concluida.

Tabla 25 Casos de prueba

Prueba. CP1: ARCE→ SIGAME

En este caso de prueba se describe la funcionalidad necesaria para enviar la información desde el SIGE ARCE a SIGAME. A su vez, este caso de prueba engloba diferentes pruebas que iremos desarrollando.

En el caso de ser pruebas similares, como por ejemplo, realizar un alta de emergencia desde distintos países, se tratarán como una prueba única aunque contenga un alta de emergencia.

En la siguiente imagen podemos ver todos los casos en los que se dividirá el caso de prueba

"ARCE → SIGAME":

La primera prueba será la realización del alta de la emergencia desde ARCE, concretamente lo podemos ver en la siguiente tabla:

CP 1.1 : ALTA DE EMERGENCIAS	
Descripción	Se realiza el traspaso de las altas de emergencias desde ARCE.
Prerrequisitos	Los roles de los usuarios que realizan las altas serán "1", "41" o "42". Se traspasarán las emergencias con estado "A" (emergencia abierta). En SIGAME la emergencia será recibida por el usuario DGPCE, quien será el encargado de hacer que las comunidades autónomas vean las emergencias.
Datos de entrada	Alta emergencia desde Ecuador (Emergencia 1)
Datos de salida	Vista desde SIGAME de las altas de las emergencias (Emergencia 1) de Ecuador.
Conclusiones	Envío de información realizado correctamente.

Tabla 26 Caso de Prueba 1.1 Alta emergencias

En los prerrequisitos de esta prueba hacemos referencia a los roles de los usuarios que realizan el alta, esto se debe a que los usuarios que, en situación de emergencia, están habilitados son (N4a y N4b) para crear la situación de emergencia y las actualizaciones. También serán los encargados de formular las peticiones de ayuda exterior y negociar las aportaciones de su país con un organismo asociado demandante.

El siguiente caso de prueba será el alta en la solicitud de recursos y lo vemos en la siguiente tabla:

CP 1.2 : ALTA DE SOLICITUD DE RECURSOS	
Descripción	Se realiza el traspaso de las solicitudes de recursos de las emergencias desde ARCE.
Prerrequisitos	La emergencia a la que haga referencia esta petición de recursos debe existir en el sistema de Emergencia ARCE. En SIGAME la solicitud de recursos será recibida por el usuario DGPCE, quien será el encargado de hacer que las comunidades autónomas vean las emergencias.
Datos de entrada	Alta petición de recursos desde Ecuador asociados a la Emergencia 1
Datos de salida	Vista desde SIGAME de las altas de las peticiones de recursos (para la Emergencia 1) de Ecuador.
Conclusiones	Envío de información realizado correctamente.

Tabla 27 Caso de prueba 1.2 Alta solicitud de recursos

Continuamos con la prueba 1.3 que mostrará Aportaciones de recursos:

CP 1.3 : ALTA DE APOTACIÓN DE RECURSOS	
Descripción	Se realiza el traspaso de las aportaciones de recursos de las emergencias desde ARCE.
Prerrequisitos	La emergencia a la que haga referencia esta aportación de recursos debe existir en el sistema de Emergencia ARCE. En SIGAME la aportación de recursos será recibida por el usuario DGPCE como una oferta, quien será el encargado de hacer que las comunidades autónomas vean las emergencias.
Datos de entrada	Alta aportación de recursos desde Guatemala para Ecuador asociados a la Emergencia.
Datos de salida	Vista desde SIGAME de las altas de las aportaciones de recursos únicamente los ofertados por Ecuador a Guatemala.

Conclusiones	Envío de información realizado correctamente, a SIGAME solo llegarán las aportaciones que realiza Ecuador puesto que son las únicas aprobadas.

Tabla 28 Caso de Prueba 1.3 Alta aportación de recursos

En esta prueba (1.3) podrían existir tres vertientes, las aportaciones aprobadas, las pendientes, y las denegadas. A SIGAME únicamente llegan las aportaciones aprobadas, ya que deben de aceptarse para cambiar de estado. En el momento en el que se realiza una aportación, el recurso toma un estado con valor "pendiente" hasta que el país afectado aprueba la aportación, en cuyo caso pasa a ser "aprobada", si la deniega el estado también cambia, a "denegada" y en caso de no ser respondida la aportación continua con el estado "pendiente". Comprobamos que se realiza el pase de información correctamente.

Una vez acabamos estas pruebas podemos decir que el traspaso de ARCE a SIGAME funciona correctamente.

A continuación, vemos el caso de prueba 2 que tratará de la migración de datos desde SIGAME hacia ARCE, en la siguiente imagen podemos ver todos los casos en los que se dividirá el caso de prueba "SIGAME → ARCE":

En la siguiente tabla comenzamos con las pruebas de la comunicación inversa, es decir, con el "Caso de Prueba 2: SIGAME → ARCE". Este caso de prueba al igual que el anterior engloba pruebas, que nos harán más fácil la comprobación de la funcionalidad del caso.

CP 2.1: ALTA DE EMERGENCIAS	
Descripción	Se realiza el traspaso de las altas de emergencias desde SIGAME.
Prerrequisitos	La comunidad que realiza el alta de las emergencias que se envían a Sistema ARCE será "DGPCE". Se traspasarán las emergencias con estado "Abierto" (emergencia abierta). En ARCE la emergencia será asociada a España, es decir el usuario con e que comprobaremos esta prueba será "localpes". Únicamente en ARCE se admite una emergencia por país, es decir, solo enviamos una emergencia de España.
Datos de entrada	Alta emergencia desde España (desde la DGPCE)
Datos de salida	Vista desde ARCE del alta de la emergencias de España.
Conclusiones	Envío de información realizado correctamente.

Tabla 29 Caso de Prueba 2.1 Alta de emergencia

La siguiente prueba será el alta en la solicitud de recursos y lo vemos en la siguiente tabla:

CP 2.2: ALTA DE SOLICITUD DE RECURSOS	
Descripción	Se realiza el traspaso de las solicitudes de recursos de la emergencia desde SIGAME.
Prerrequisitos	La emergencia a la que haga referencia esta petición de recursos debe existir en el sistema de Emergencia SIGAME. En ARCE la solicitud de recursos será recibida por el usuario "localpes".
Datos de entrada	Alta petición de recursos desde España asociados a la emergencia
Datos de salida	Vista desde ARCE de las altas de las peticiones de recursos para la emergencia de España.
Conclusiones	Envío de información realizado correctamente.

Tabla 30 Caso de prueba 2.2 Alta solicitud de recursos

Continuamos con la prueba 2.3 que mostrará Aportaciones de recursos:

CP 2.3: ALTA DE APORTACIÓN DE RECURSOS	
Descripción	Se realiza el traspaso de las aportaciones de recursos de las emergencias desde SIGAME.
Prerrequisitos	La emergencia a la que haga referencia esta aportación de recursos debe existir en el sistema de Emergencia SIGAME, pasarán las aportaciones que hacen las comunidades a España. El campo "tipo" de la oferta determinará el lugar de entrega dependiendo de este campo se obtendrán los lugares de entrega de la oferta o de la solicitud de recursos. Este lugar de entrega debe seguir un determinado patrón. Suponemos que está correcto.
Datos de entrada	Alta aportación de recursos desde la comunidad de Madrid a la emergencia de la DGPCE.
Datos de salida	Vista desde ARCE de las altas de las aportaciones de recursos.
Conclusiones	Envío de información realizado correctamente, a ARCE solo llegarán las aportaciones que realiza España que son las únicas aprobadas.

Tabla 31 Caso de Prueba 3.3 Alta aportación de recursos

En esta prueba (2.3) existen dos tipos, las aportaciones que se envían son 2:

- Aportación de recursos de la comunidad de Madrid a la DGPCE para solventar la Emergencia.
- Aportación de recursos a los países Iberoamericanos, es decir, los que se encuentran en ARCE que no son España.

Sin embargo, a ARCE únicamente llega el informe de las aportaciones de España a España, esto es porque en la DGPCE no posee recursos. La aplicación SIGAME no deja que la DGPCE oferte.

5.2 Análisis de resultados

Una vez hemos ejecutado los pasos anteriores podemos observar que la migración de datos se realiza correctamente, tal y como se muestra en el video adjunto. De los casos de prueba realizados en el apartado anterior (5.1 Proceso de evaluación), se han producido las siguientes conclusiones. Hemos observado que a la hora de tratar con recursos, tanto solicitados como aportados (ofertados), en los dos sistemas no coinciden, es decir, el catálogo de recursos en un futuro debería ser igual, lo mismo ocurre con lo referente a los lugares.

CASO DE PRUEBA	CONCLUSIÓN
CP 1: ARCE → SIGAME	
Prueba 1.1	OK
Prueba 1.2	OK
Prueba 1.3	OK
CP 2: SIGAME → ARCE	
Prueba 2.1	OK
Prueba 2.2	OK
Prueba 2.3	OK

Tabla 32 Control de resultados

6 Conclusión

En este capítulo se presentan las conclusiones obtenidas del desarrollo de este proyecto y se ofrecen algunas sugerencias para una posible ampliación futura. Esta sección se dividirá en cuatro apartados, el primero, **Aportaciones realizadas,** se referirá a lo que hemos aportado con la realización del proyecto a los SIGE SIGAME y ARCE. En el segundo apartado, **Trabajos futuros,** nos centraremos en las mejoras que podemos incluir para que su funcionamiento sea más eficaz. En el tercer punto, **Problemas encontrados**, se hablará sobre los problemas que se han encontrado a la hora de la realización de este proyecto. Y por último, veremos las **Opiniones personales** donde se comentará de forma particular las impresiones que se han obtenido con la realización del proyecto.

6.1 Aportaciones realizadas

La necesidad de manejar con eficacia situaciones de la emergencia ha crecido desde hace pocos años, debido al aumento del número y las consecuencias de catástrofes y desastres.

A lo largo de la realización de este proyecto hemos conseguido la coordinación entre los sistemas de emergencia SIGAME y ARCE. Hemos logrado la interoperabilidad entre distintos esquemas de metadatos, a través de un mapeo de los datos de ambos sistemas, y el establecimiento de correspondencias entre informaciones en diferentes formatos para la conversión de elementos de metainformación que permite hacerlos compatibles.

La razón de esta alianza entre ambos sistemas es la intención de compartir información sobre el estado de las emergencias, las solicitudes, las aportaciones,… esto hará mas fácil y rápido el hecho de solucionar una emergencia. Esta intención de compartir información, requiere de la colaboración entre los distintos organismos y entidades. Esta colaboración implica el intercambio de información entre los participantes en el proceso de gestión. ARCE y SIGAME son sistemas de gestión de emergencias independientes pero susceptibles de trabajar en común, motivo por el cual es necesario proporcionar mecanismos de interoperabilidad entre los mismos.

Se trata de sistemas independientes entre sí pero con objetivos similares, lo que invita al uso

combinado de ambos. Este hecho requiere de mecanismos de intercambio de información.

6.2 Trabajos futuros

Con la finalización de todo el proyecto quedan cuestiones pendientes de un mayor desarrollo y se ven nuevas líneas para seguir una evolución futura.

Desde el punto de vista del diseño, en una primera versión del proyecto, se realizó una estructura diferente a la entregada, esta estructura se realizaba a través de un mapeo de datos que pasaba por tres documentos XML, existía un documento asociado a cada uno de los sistemas que reflejaba fielmente las estructuras de almacenamiento de datos, y aparte de estos dos documentos, entre ellos se obtenía otro fichero XML común a ambos, esta solución almacenaba información redundante y aumentaba el tiempo de ejecución del programa, así que se decidió que para esta última versión se realizaran todas las tareas que descubrían estos tres documentos en uno solo, de tal manera que migrara la información de un sistema al documento XML cuando lo decidieran los países afectados, y de este documento se enviaría por FTP al sistema contrario. En un futuro esta solución también se podría mejorar, ahora mismo el servidor espera una petición, la petición cuando llega se resuelve y espera la siguiente, se podrían realizar las tareas necesarias para que se procesaran varias peticiones a la vez.

Desde las aplicaciones considero que se podrían mejorar algunas cosas, como por ejemplo, crear un catálogo de recursos común a ambos sistemas de emergencia, puesto que no se corresponde, nosotros en el proyecto buscamos que exista el nombre del recurso que se solicita u oferta, en caso de no existir no lo incluimos ya que puede existir con otro nombre y se duplicarían datos, así que en ese caso no se insertaría el recurso solicitado. Este mismo problema ocurre con los lugares, evidentemente en SIGAME no aparecen los lugares asociados, por ejemplo, a Ecuador, ya que no tendría sentido pero para que este proyecto fuera más consistente se podría crear.

El usuario DGPCE toma un papel muy importante en SIGAME ya que todas las emergencias que llegan de ARCE se asocian a este usuario, de tal manera que sus acciones de forma indirecta se ven sobrecargadas, una solución a esto sería crear un usuario que se encargara de las emergencias que llegan de ARCE, y gestionara los recursos que se solicitan y los que se ofertan en cada momento.

Desde el punto de vista del código fuente, estoy segura que este se podría optimizar, ya que al iniciar el proyecto yo no había tenido ningún contacto con el lenguaje Java.

6.3 Problemas encontrados

En todo este tiempo se han encontrado ciertas complicaciones a la hora de realizar el proyecto, en un principio surgieron algunos problemas al elegir el lenguaje de programación en el cual iba a estar codificado el proyecto, ya que nunca había programado con ningún lenguaje con orientación a objetos, la documentación que emplee en el capítulo 2 .2 Estado de la cuestión, me ayudó a entender este tipo de programación y otros muchos conceptos que no había estudiado hasta entonces, como el envío de ficheros mediante FTP, la estructura cliente-servidor,…

Al estudiar los sistemas con los que iba a trabajar también me surgieron dudas, en un principio no tenía ningún tipo de documentación de los sistemas, estos sistemas son estructuralmente muy diferentes, ya no solo por el lenguaje en el cual están implementadas las bases de datos, sino también por el tratamiento que se le da a los datos y las estructuras que los almacenan, un ejemplo para este caso pueden ser las fechas, en SIGAME son mucho más flexibles; sin embargo, en ARCE están codificadas bajo un formato de tipo fecha, que me han obligado a desarrollar método de conversión entre ambos para su tratamiento. También se observa que en el sistema SIGAME no existe una integridad referencial entre sus tablas, y otros problemas ya comentados en el punto anterior, el catálogo de recursos no es el mismo ni los lugares se identifican los unos con los otros.

6.4 Opiniones personales

Este proyecto me ha servido para aprender a abstraerme y a desarrollar en un lenguaje nuevo para mí, que laboralmente me está abriendo puertas, ya que en la actualidad me dedico a desarrollo de aplicaciones en lenguaje Java. Además de conocimientos de programación me ha aportado una visión distinta de la carrera cursada, durante los años de carrera he aprendido mucho, pero la mayoría de las materias se quedaban en la teoría, hoy después de todo este tiempo he puesto en práctica todo aquello que nos explicaron sobre la gestión de proyectos y lo importante que era un buen estudio y planificación de las tareas que queríamos desarrollar.

Sin embargo, he de decir que hubo un tiempo que el proyecto fue difícil, empleaba muchas horas y los avances eran pocos, pero a lo largo del tiempo me he dado cuenta de que todo ese esfuerzo ha merecido la pena. En este momento, una vez finalizado el proyecto se observan algunas conclusiones sobre cómo se ha abordado el tema que hoy se ven obvias, pero al inicio y con todo el trabajo por hacer no se veían claramente. Desde el primer diseño hasta el diseño definitivo han pasado muchas etapas, y puedo decir que las fases de análisis y diseño han sido las más importantes. El principal aporte a nivel personal que me ha dado este proyecto ha sido tener verdadera conciencia de cómo funcionan realmente los procesos de desarrollo software, sobre todo cuando se refiere a incluir una funcionalidad nueva a sistemas ya existentes: cómo se puede, paso a paso, interpretar y abstraer, un problema de la vida real y modelar una solución software adecuada a dicho problema, empleando para ello todas las herramientas de software y de conocimiento que existen en el ámbito de la Ingeniería del software. En resumen, una visión más global de lo que un proyecto software y un autentico entendimiento de qué función y razón de ser tiene cada paso.

Durante todo el proyecto me ha parecido interesante la temática del proyecto, el haberme informado sobre los sistemas de Emergencia y el hecho de intentar aportar algo a aplicaciones estables, me resultaba un poco intenso pero a su vez interesante, ya que estos sistemas de gestión de emergencia en mi opinión son realmente útiles. El hecho de ver que en estos casos una verdadera buena gestión de una situación de emergencia puede ser vital, y ofertar recursos a países con problemas le pueden facilitar mucho las cosas, y no solo eso sino que también si se agiliza la gestión.

7 Bibliografía

[1] Del Castillo, Álvaro. "Zope el servidor de aplicaciones libre". [Online]. (Mayo 2009) Available from World Wide Web: http://www.programacion.com/tutorial/zope/

[2] Grupo de Bases de Datos Avanzadas. Universidad Carlos III de Madrid. "Conceptos y objetivos de las Bases de Datos". [online]. (Mayo 2009). Available from World Wide Web: http://basesdatos.uc3m.es/fileadmin/Docencia/FuBD/Teoria/Tema10809.pdf

[3] Anónimo. "Base de datos relacional". [Online]. (Mayo 2009). Available from World Wide Web: http://es.wikipedia.org/wiki/Bases_de_datos_relacionales

[4] Davis, Michele E.; Phillips, Jon A. "Manual EasyPhp 1.8". (Ed. Anaya Multimedia) (2000).

[5] The PostgreSQL Global Development Group, "PostgreSQL 8.3.7 Documentation". [Online]. (Mayo 2009) Available from World Wide Web:

http://www.postgresql.org/docs/current/static/

[6] XML Core Working Group Public Page. "Extensible Markup Language (XML)". [Online]. (Mayo 2009) Available from World Wide Web:

http://www.w3.org/XML/

http://www.w3.org/XML/Schema

Dan Livingstone. "Guía Esencial XML". (Ed. Prentice Hall). 2002.

[7] García de Jalón, Javier; Rodríguez, José Ignacio; Mingo, Iñigo; Imaz, Aitor; Brazález, Alfonso; Larzabal, Alberto, Calleja, Jesús, García, Jon. "Aprenda Java como si estuviera en primero". (publicado en la Web). Universidad de Navarra. 2000. (Mayo 2009) Available from World Wide Web:

http://www.tecnun.es/asignaturas/Informat1/ayudainf/aprendainf/Java/Java2.pdf

Sun Microsystems. "JDK 6 Documentation". [Online]. (Mayo 2009) Available from World Wide Web: http://java.sun.com/javase/6/docs/

[8] Sun Microsystems. "Java SE Technologies - Database". [Online]. (Mayo 2009) Available from World Wide Web: http://java.sun.com/jdbc/

Kevin Mukhar; Todo Lauinger; John Carnell, "Bases de datos con Java". Ediciones Anaya, 2001.

Sun. Traductor: Palos, Juan Antonio. "El API JAXB". [Online]. (Mayo 2009) Available from World Wide Web: http://www.programacion.com/tutorial/jaxb/

http://java.sun.com/developer/technicalArticles/WebServices/jaxb/

[9] Cioroinu, Andrei; Akif, Mohammad; Brodhead, Steven. "Java y XML". (Anaya multimedia). 1ª edición. 2002.

[10] Gracia, Joaquín. "UML: Casos de Uso. Use case". [online]. (Mayo 2009) Available from World Wide Web:

http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php

[11] Ministerio de Administraciones Públicas. "Metodología MÉTRICA Versión 3 ". [online]. (Mayo 2009) Available from World Wide Web:

http://www.csi.map.es/csi/metrica3/gespro.pdf

Anónimo. "Base de datos relacional". [online]. (Mayo 2009). Available from World Wide Web: http://es.wikipedia.org/wiki/Gesti%C3%B3n_de_proyectos

[12] Gómez Gallego, Juan Pablo. "Fundamentos de la Metodología RUP. Rational Unified Process". (16/09/2007). Universidad Tecnológica de Pereira. [online]. (Mayo 2009). Available from World Wide Web: www.scribd.com/doc/297224/RUP

[13] Universidad de Valladolid. "Documento de análisis de Requisitos". [online]. (Mayo 2009). Available from World Wide Web:

http://www.infor.uva.es/~jmmc/ingsoft/docs/issrad.pdf

[14] r_niella. "Capítulo 1: Cliente servidor". [online]. (Mayo 2009). Available from World Wide Web: http://ar.geocities.com/r_niella/Document/t_cap1.htm

[15] Oré B., Alexander. "¿Cómo realizar pruebas funcionales?". [online]. (Mayo 2009). Available from World:

http://www.calidadysoftware.com/testing/como_realizar_pruebas_funcionales.php

Anexo I. Control de versiones

En este apartado se realiza un seguimiento de las versiones de la memoria. Por cada versión se indicará el identificador de versión, la fecha de finalización y una breve descripción de su contenido, en el contenido se expresa en dos apartados, la preparación serán los puntos nuevos a tratar y las correcciones serán los puntos revisados por el tutor que tienen errores.

TABLA DE MODIFICACIONES RELEVANTES		
Versión	Fecha	Avance y modificaciones relevantes
1.0	22/12/2008	• Cambio de formato Preparación de los apartados: • Introducción. • Estudio del problema.
2.0	10/01/2009	Corrección de los apartados: • Introducción. • Estudio del problema. Preparación de los apartados: • Gestión de software. • Solución. • Evaluación. • Conclusiones.

3.0	15/03/2009	Corrección de los apartados: • Gestión de software. • Solución. • Evaluación. Preparación de los apartados: • Introducción. • Solución. • Evaluación. • Conclusiones
4.0	15/04/2009	Corrección de los apartados: • Solución. Preparación de los apartados: • Solución. • Evaluación. • Conclusiones
5.0	04/05/2009	Corrección de los apartados: • Solución. • Evaluación. Preparación de los apartados: • Solución. • Conclusiones

6.0		Corrección de los apartados: • Evaluación • Conclusiones. • Bibliografía. Preparación de los apartados: • Solución. • Evaluación. • Conclusiones
7.0		Revisión del documento previa a la entrega al tribunal. Cambios: Pequeñas modificaciones en formato y actualización de ciertas referencias a imágenes y tablas, bibliografía.

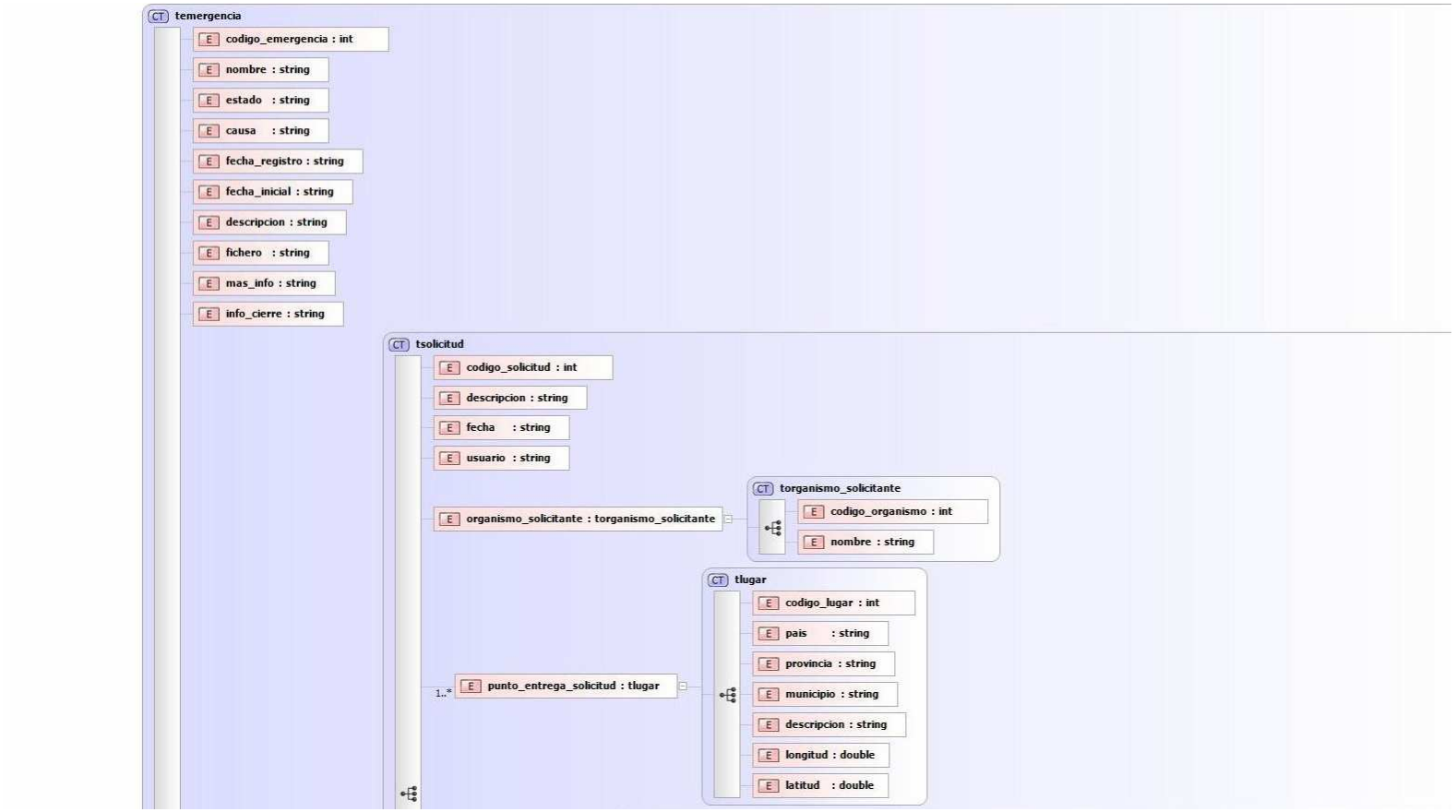
Anexo II. Modelo de Información

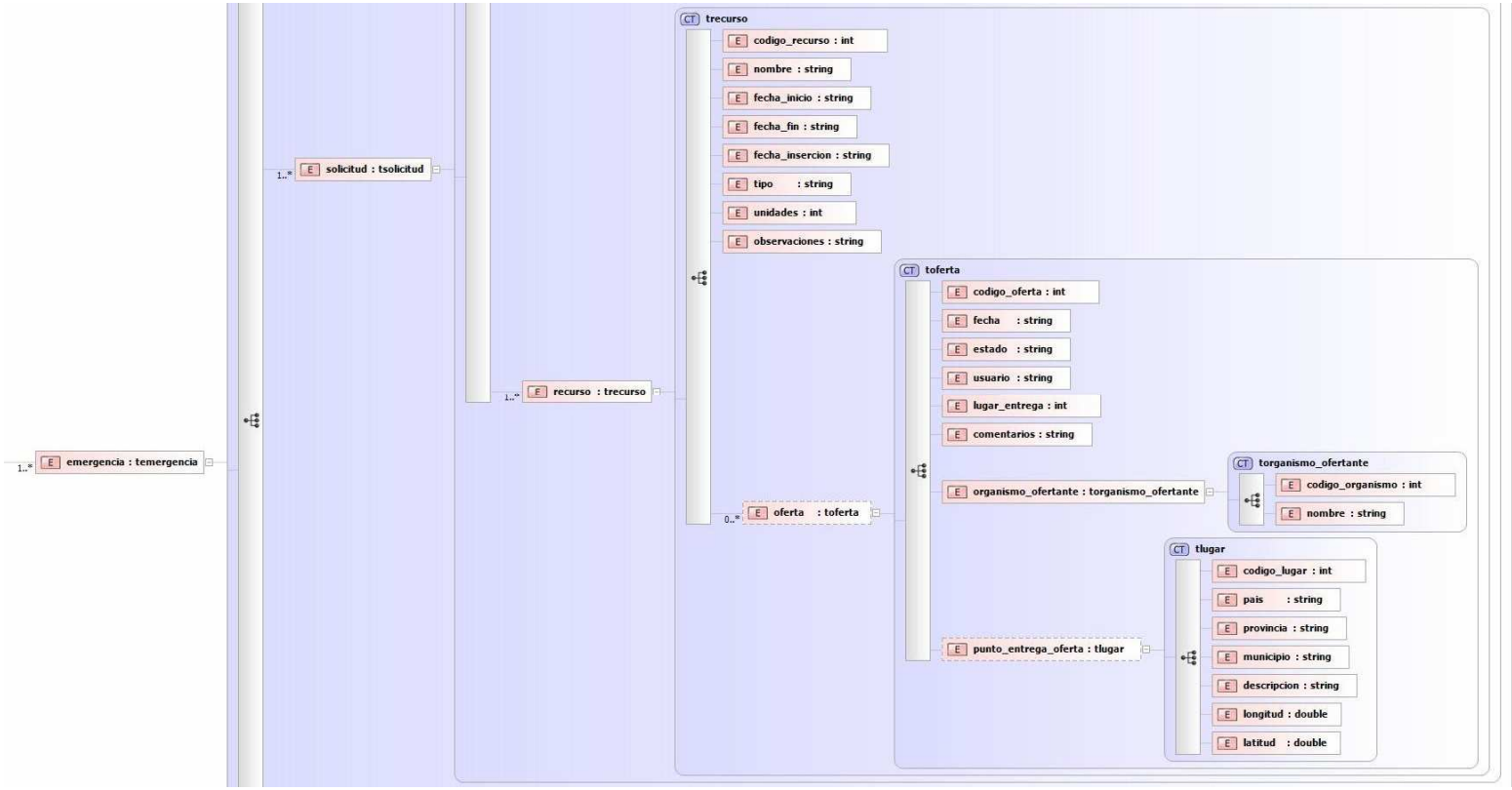
En este anexo se verá la información que almacena el documento XML una vez se ha ejecutado el programa. En primer lugar mostraremos una imagen donde se pueden ver los tipos de datos de la estructura. A continuación, se exponen los datos que se obtienen al leer de un sistema de emergencia, primero se mostrará el documento formado por las emergencias leídas de ARCE y seguidamente las leídas desde SIGAME.

En la siguiente imagen podemos ver la representación de datos que almacenará el esquema XML que se ha definido acorde con el diagrama de clases visto en la sección *Definición del Modelo de Información,* en el Capítulo 4. Solución. Observamos que una emergencia de tipo Temergencia guarda todos los datos referentes a la misma, contiene las solicitudes de medios, el lugar donde ocurre, los daños materiales que ha causado y las víctimas afectadas en dicha emergencia. Cada una de las solicitudes contendrá datos concretos de la misma y además, los datos del organismo solicitante, el lugar donde se solicitan los medios y los recursos que se precisarán, para cada uno de los recursos, se podrán almacenar datos concretos de este recurso y las ofertas que tenga asociado este recurso, en caso de tenerlas. Para cada oferta, si tuviese, se registran los datos de esta oferta, junto con la información del organismo ofertante y el lugar de entrega asociado a esta oferta. En el fichero podrán existir varias emergencias, dentro del elemento emergencias_pfc.

Para obtener la siguiente imagen se ha utilizado el programa Liquid Xml.

En cada momento, el fichero XML puede tomar varios valores, a continuación vemos los datos que se guardan al leer del Sistema de Emergencia SIGAME e insertar en ARCE y viceversa.


temergencia
codigo_emergencia : int
nombre : string
estado : string
causa : string
fecha_registro : string
fecha_inicial : string
descripcion : string
fichero : string
mas_info : string
info_cierre : string
tsolicitud
codigo_solicitud : int
descripcion : string
fecha : string
usuario : string
organismo_solicitante : torganismo_solicitante
torganismo_solicitante
codigo_organismo : int
nombre : string
1..*
punto_entrega_solicitud : tlugar
tlugar
codigo_lugar : int
pais : string
provincia : string
municipio : string
descripcion : string
longitud : double
latitud : double


emergencia : temergencia
solicitud : tsolicitud
recurso : trecurso
trecurso
codigo_recurso : int
nombre : string
fecha_inicio : string
fecha_fin : string
fecha_insercion : string
tipo : string
unidades : int
observaciones : string
oferta : toferta
toferta
codigo_oferta : int
fecha : string
estado : string
usuario : string
lugar_entrega : int
comentarios : string
organismo_ofertante : torganismo_ofertante
punto_entrega_oferta : tlugar
torganismo_ofertante
codigo_organismo : int
nombre : string
tlugar
codigo_lugar : int
pais : string
provincia : string
municipio : string
descripcion : string
longitud : double
latitud : double


tlugar
codigo_lugar : int
pais : string
provincia : string
municipio : string
descripcion : string
longitud : double
latitud : double
lugar : tlugar
tdannos_materiales
dannos_viviendas : string
dannos_edi_ofi : string
dannos_hospitales : string
dannos_agua : string
dannos_electricidad : string
dannos_telefonia : string
dannos_carreteras : string
dannos_alcantarillado : string
dannos_alimentos : string
dannos_sanitaria : string
cor_carreteras : string
cor_ferrocarril : string
cor_aeropuertos : string
cor_puertos : string
dannos_materiales : tdannos_materiales
tvictimas
afectados : int
fallecidos : int
heridos : int
desaparecidos : int
evacuados : int
atrapados : int
aislados : int
victimas : tvictimas

Para el caso de prueba 1 desarrollado en el Capítulo 5. Evaluación, podemos ver como se produce el alta de una emergencia en ARCE, por parte de Ecuador, también se observa que Guatemala realiza una aportación de medios. En este caso se pretende que SIGAME sea capaz de registrar esa información, es decir, cualquier usuario de SIGAME debería de poder acceder a esta información, tanto a la emergencia como a los recursos que ya están aportados.

Seguidamente, mostraremos la información que queda almacenada en la estructura XML al ejecutar el programa. Los identificadores no se deben tener en cuenta, ya que son los que leemos directamente de la base de datos. La almacenada en el XML será la siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<emergencias_pfc xmlns="http://xml.netbeans.org/examples/targetNS">
    <emergencia>
        <codigo_emergencia>33</codigo_emergencia>
        <nombre>Incendio forestal</nombre>
        <estado>A</estado>
        <causa>Incendio provocado por una hoguera mal apagada.&#xD;</causa>
        <fecha_registro>2009-05-03 17:12:04+02</fecha_registro>
        <fecha_inicial>2009-05-03 17:09:00+02</fecha_inicial>
        <descripcion>Incendio provocado por una hoguera mal apagada.&#xD;</descripcion>
        <fichero></fichero>
        <mas_info>Se necesitarÃ¡n ATS principalmente, esta informaciÃ³n se encuentra disponible en los recursos solicitados.</mas_info>
        <solicitud>
            <codigo_solicitud>50</codigo_solicitud>
            <fecha>2009-05-03 17:20:50+02</fecha>
            <usuario>localpec</usuario>
            <punto_entrega_solicitud>
                <codigo_lugar>282</codigo_lugar>
                <pais>9</pais>
                <provincia></provincia>
```

<municipio>Quito</municipio>

<descripcion>Aeropuerto Mariscal Sucre, icao: SEQU</descripcion>

<longitud>0.0</longitud>

<latitud>0.0</latitud>

</punto_entrega_solicitud>

<recurso>

<codigo_recurso>3</codigo_recurso>

<nombre>Especialistas en protecciÃ³n civil</nombre>

<fecha_inicio>2009-05-03 17:20:50+02</fecha_inicio>

<fecha_insercion>2009-05-03 17:20:50+02</fecha_insercion>

<unidades>15</unidades>

<observaciones>Persona con la formaciÃ³n y experiencia adecuadas para poder asesorar tÃ©cnicamente y colaborar en algunas de las actuaciones de prevenciÃ³n o de emergencia caracterÃ-sticas de protecciÃ³n civil.</observaciones>

</recurso>

<recurso>

<codigo_recurso>26</codigo_recurso>

<nombre>A.T.S.</nombre>

<fecha_inicio>2009-05-03 17:20:50+02</fecha_inicio>

<fecha_insercion>2009-05-03 17:20:50+02</fecha_insercion>

<unidades>40</unidades>

<observaciones>Diplomado Universitario en EnfermerÃ-a</observaciones>

<oferta>

<codigo_oferta>22</codigo_oferta>

<fecha>2009-05-03 17:22:58+02</fecha>

<estado>cedido</estado>

<lugar_entrega>18</lugar_entrega>

<comentarios>Los miembros cedidos completan los recursos solicitados por Ecuador de ATS y Cruz Roja.</comentarios>

<punto_entrega_oferta>

<codigo_lugar>18</codigo_lugar>

<pais>11</pais>

<provincia></provincia>

<municipio>Palma, La</municipio>

<descripcion>Aeropuerto de La Palma, icao: GCLA, longitud: 174513W</descripcion>

<longitud>174513.0</longitud>

<latitud>283721.0</latitud>

</punto_entrega_oferta>

</oferta>

</recurso>

<recurso>

<codigo_recurso>101</codigo_recurso>

<nombre>Cruz Roja</nombre>

<fecha_inicio>2009-05-03 17:20:50+02</fecha_inicio>

<fecha_insercion>2009-05-03 17:20:50+02</fecha_insercion>

<unidades>10</unidades>

<observaciones>OrganizaciÃ³n internacional de ayuda humanitaria. Unidad de medida: sanitarios.</observaciones>

<oferta>

<codigo_oferta>22</codigo_oferta>

<fecha>2009-05-03 17:22:58+02</fecha>

<estado>cedido</estado>

<lugar_entrega>18</lugar_entrega>

<comentarios>Los miembros cedidos completan los recursos solicitados por Ecuador de ATS y Cruz Roja.</comentarios>

<punto_entrega_oferta>

<codigo_lugar>18</codigo_lugar>

<pais>11</pais>

<provincia></provincia>

<municipio>Palma, La</municipio>

<descripcion>Aeropuerto de La Palma, icao: GCLA, longitud: 174513W</descripcion>

<longitud>174513.0</longitud>

<latitud>283721.0</latitud>

</punto_entrega_oferta>

```
                        </oferta>
                </recurso>
        </solicitud>
        <lugar>
                <codigo_lugar>9</codigo_lugar>
                <pais>Ecuador</pais>
                <provincia></provincia>
                <municipio></municipio>
                <longitud>0.0</longitud>
                <latitud>0.0</latitud>
        </lugar>

        <dannos_materiales/>
        <victimas>
                <afectados>4000</afectados>
                <fallecidos>100</fallecidos>
                <heridos>200</heridos>
                <desaparecidos>300</desaparecidos>
                <evacuados>400</evacuados>
                <atrapados>500</atrapados>
                <aislados>600</aislados>
        </victimas>
    </emergencia>
</emergencias_pfc>
```

Para el caso de prueba 2, el cual se exponía en el Capítulo 5. Evaluación, podemos ver como se produce el alta de una emergencia en SIGAME, en este coso es la DGPCE quien realiza la acción, puesto que es el usuario que decide cuales emergencias hay que informar en ARCE y de cuales no, también se observa que la Comunidad de Madrid realiza una oferta de medios. Ahora es ARCE quien debe ser capaz de registrar esa información, es decir, cualquier usuario de ARCE debería de poder acceder a esta información, tanto a la emergencia como a los recursos que ya están

ofertados. Al ejecutar el programa podemos ver que la información almacenada en el XML será la siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<emergencias_pfc xmlns="http://xml.netbeans.org/examples/targetNS">
    <emergencia>
        <codigo_emergencia>140</codigo_emergencia>
        <nombre>InundaciÃ³n</nombre>
        <estado>abierta</estado>
        <causa>inundaciÃ³n causada por lluvias torrenciales.</causa>
        <fecha_registro>09/05/2009 15:29</fecha_registro>
        <fecha_inicial>09/05/2009 17:27</fecha_inicial>
        <descripcion>Emergencia DGPCE Tarragona</descripcion>
        <fichero></fichero>
        <mas_info>No existe informaciÃ³n acerca de los cortes de suministro</mas_info>
        <solicitud>
            <codigo_solicitud>140</codigo_solicitud>
            <descripcion>InformaciÃ³n sobre los recursos solicitados, ATS. LUGAR DE ENTREGA: Puerto de Sant Carles de la RÃ¡pita.</descripcion>
            <fecha>09/05/2009 15:41</fecha>
            <usuario>localpes</usuario>
            <organismo_solicitante>
                <codigo_organismo>20</codigo_organismo>
                <nombre>DGPCE</nombre>
            </organismo_solicitante>
            <punto_entrega_solicitud>
                <codigo_lugar>0</codigo_lugar>
                <pais>EspaÃ±a</pais>
                <provincia>DGPCE</provincia>
                <municipio>DGPCE</municipio>
                <descripcion>DirecciÃ³n de la DGPCE</descripcion>
                <longitud>0.0</longitud>
```

<latitud>0.0</latitud>
</punto_entrega_solicitud>
<recurso>
<codigo_recurso>26</codigo_recurso>
<nombre>A.T.S.</nombre>
<fecha_inicio>11/05/2009</fecha_inicio>
<fecha_fin></fecha_fin>
<fecha_insercion>09/05/2009 15:41</fecha_insercion>
<tipo>origen</tipo>
<unidades>30</unidades>
<observaciones>Diplomado Universitario en EnfermerÃ-a</observaciones>
</recurso>
<recurso>
<codigo_recurso>26</codigo_recurso>
<nombre>A.T.S.</nombre>
<tipo>origen</tipo>
<unidades>0</unidades>
<observaciones>Diplomado Universitario en EnfermerÃ-a</observaciones>
</recurso>
</solicitud>
<solicitud>
<codigo_solicitud>140</codigo_solicitud>
<descripcion>InfornaciÃ³n sobre los recursos solicitados, bomberos. LUGAR DE ENTREGA: Puerto de Sant Carles de la RÃ pita (Tarragona)</descripcion>
<fecha>09/05/2009 15:42</fecha>
<usuario>localpes</usuario>
<organismo_solicitante>
<codigo_organismo>20</codigo_organismo>
<nombre>DGPCE</nombre>
</organismo_solicitante>
<punto_entrega_solicitud>

<codigo_lugar>0</codigo_lugar>

<pais>EspaÃ±a</pais>

<provincia>DGPCE</provincia>

<municipio>DGPCE</municipio>

<descripcion>DirecciÃ³n de la DGPCE</descripcion>

<longitud>0.0</longitud>

<latitud>0.0</latitud>

</punto_entrega_solicitud>

<recurso>

<codigo_recurso>36</codigo_recurso>

<nombre>Bomberos</nombre>

<fecha_inicio>09/05/2009</fecha_inicio>

<fecha_fin></fecha_fin>

<fecha_insercion>09/05/2009 15:42</fecha_insercion>

<tipo>origen</tipo>

<unidades>20</unidades>

<observaciones>Grupo profesional jerarquizado de personas con formaciÃ³n, medios y experiencia adecuados para llevar a cabo labores de extinciÃ³n de incendios y salvamento</observaciones>

</recurso>

<recurso>

<codigo_recurso>36</codigo_recurso>

<nombre>Bomberos</nombre>

<fecha_inicio>09/05/2009</fecha_inicio>

<fecha_fin></fecha_fin>

<fecha_insercion>09/05/2009 15:42</fecha_insercion>

<tipo>origen</tipo>

<unidades>20</unidades>

<observaciones>Grupo profesional jerarquizado de personas con formaciÃ³n, medios y experiencia adecuados para llevar a cabo labores de extinciÃ³n de incendios y salvamento</observaciones>

<oferta>

<codigo_oferta>84</codigo_oferta>

<fecha>10/05/2009 09:40</fecha>

```
                                <estado>Cedido</estado>
                                <usuario>InformaciÃ³n de la cesiÃ³n de los bomberos.</usuario>
                                <lugar_entrega>0</lugar_entrega>
                                <comentarios>InformaciÃ³n del Lugar de entrega de los Bomberos. LUGAR DE ENTREGA: Puerto de Sant Carles de la RÃ pita (Tarragona).</comentarios>
                                <organismo_ofertante>
                                        <codigo_organismo>11</codigo_organismo>
                                        <nombre>Comunidad de Madrid</nombre>
                                </organismo_ofertante>
                                <punto_entrega_oferta>
                                        <codigo_lugar>0</codigo_lugar>
                                        <pais>EspaÃ±a</pais>
                                        <provincia>DGPCE</provincia>
                                        <municipio>DGPCE</municipio>
                                        <descripcion>DirecciÃ³n de la DGPCE</descripcion>
                                        <longitud>0.0</longitud>
                                        <latitud>0.0</latitud>
                                </punto_entrega_oferta>
                        </oferta>
                </recurso>
        </solicitud>
        <lugar>
                <codigo_lugar>0</codigo_lugar>
                <pais>Spain</pais>
                <provincia></provincia>
                <municipio></municipio>
                <descripcion></descripcion>
                <longitud>0.0</longitud>
                <latitud>0.0</latitud>
        </lugar>
        <dannos_materiales>
                <dannos_viviendas></dannos_viviendas>
```

```
                <dannos_edi_ofi></dannos_edi_ofi>
                <dannos_agua></dannos_agua>
                <dannos_electricidad></dannos_electricidad>
                <dannos_telefonia></dannos_telefonia>
                <dannos_carreteras>3</dannos_carreteras>
                <cor_carreteras>3</cor_carreteras>
                <cor_ferrocarril></cor_ferrocarril>
                <cor_aeropuertos></cor_aeropuertos>
                <cor_puertos>2</cor_puertos>
            </dannos_materiales>
            <victimas>
                <afectados>0</afectados>
                <fallecidos>2</fallecidos>
                <heridos>200</heridos>
                <desaparecidos>0</desaparecidos>
                <evacuados>0</evacuados>
                <atrapados>0</atrapados>
                <aislados>0</aislados>
            </victimas>
        </emergencia>
    </emergencias_pfc>
```

Anexo III. Modelo de datos de partida

Las tablas que forman la Base de Datos de SIGAME son las siguientes:

	Tabla	Acción	Registros	Tipo	Cotejamiento	Tamaño	Residuo a depurar
☐	alertas_mensajes		0	MyISAM	latin1_swedish_ci	1.0 KB	-
☐	alertas_noticias		0	MyISAM	latin1_swedish_ci	1.0 KB	-
☐	alertas_solicitudes		40	MyISAM	latin1_spanish_ci	3.2 KB	-
☐	catalogo_recursos		449	MyISAM	latin1_spanish_ci	85.9 KB	-
☐	comunidades		20	MyISAM	latin1_swedish_ci	2.4 KB	-
☐	contactos		21	MyISAM	latin1_swedish_ci	4.4 KB	-
☐	encuestas		0	MyISAM	latin1_swedish_ci	1.0 KB	-
☐	envio_mensaje		0	MyISAM	latin1_swedish_ci	1.0 KB	-
☐	estado_evolucion		2	MyISAM	latin1_swedish_ci	2.5 KB	-
☐	evento		93	MyISAM	latin1_swedish_ci	13.4 KB	-
☐	evolucion		0	MyISAM	latin1_swedish_ci	2.1 KB	88 Bytes
☐	fuentes_rss		5	MyISAM	latin1_swedish_ci	2.4 KB	-
☐	incendio		0	MyISAM	latin1_swedish_ci	1.0 KB	-
☐	incidencias		0	MyISAM	latin1_swedish_ci	1.0 KB	-
☐	mensaje		0	MyISAM	latin1_swedish_ci	1.0 KB	-
☐	noticias		1	MyISAM	latin1_swedish_ci	2.0 KB	-
☐	oferta		8	MyISAM	latin1_swedish_ci	5.6 KB	1,032 Bytes
☐	organismos		0	MyISAM	latin1_swedish_ci	1.0 KB	-
☐	provincias		54	MyISAM	latin1_swedish_ci	3.1 KB	-
☐	recurso		24	MyISAM	latin1_swedish_ci	6.1 KB	-
☐	recursos_cedidos		1	MyISAM	latin1_swedish_ci	2.1 KB	-
☐	recursos_solicitados		5	MyISAM	latin1_swedish_ci	3.2 KB	-
☐	relaciones_recursos		450	MyISAM	latin1_swedish_ci	18.2 KB	-
☐	solicitud		2	MyISAM	latin1_spanish_ci	2.7 KB	-
☐	solicitudcerrada		0	MyISAM	latin1_spanish_ci	1.0 KB	-
☐	usuarios		5	MyISAM	latin1_swedish_ci	2.7 KB	-
	26 tabla(s)	Número de filas	1,180	MyISAM	latin1_swedish_ci	170.9 KB	1.1 KB

Ilustración 39 Tablas Base de datos SIGAME

Las tablas que nos interesan de la Base de Datos SIGAME son cuatro activamente, es decir, al dar de alta una emergencia, o una oferta se modificarán las siguientes tablas:

Tabla Solicitud. Esta tabla se encarga de almacenar datos referentes a la "Solicitud", es decir, los campos que se rellenarán serán los que faciliten información sobre la fecha en la que se da de alta la emergencia, descripción, lugar,…

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	id	int(11)			No		auto_increment	
☐	descripcion	longtext	latin1_spanish_ci		No			
☐	fecha	varchar(19)	latin1_spanish_ci		No	00/00/0000 00:00:00		
☐	comunidad	varchar(25)	latin1_spanish_ci		No			
☐	detalles	longtext	latin1_spanish_ci		Sí	*NULL*		
☐	para	varchar(50)	latin1_spanish_ci		No			
☐	estado	varchar(25)	latin1_spanish_ci		No			
☐	fechaInsercionSolicitud	varchar(19)	latin1_spanish_ci		No	00/00/0000 00:00:00		
☐	incidencia	varchar(50)	latin1_spanish_ci		No	generica		
☐	pais	varchar(50)	latin1_spanish_ci		No	Spain		
☐	provincia	varchar(50)	latin1_spanish_ci		No			
☐	municipio	varchar(50)	latin1_spanish_ci		No			
☐	lugar	varchar(50)	latin1_spanish_ci		No			
☐	longitud	varchar(50)	latin1_spanish_ci		No			
☐	latitud	varchar(50)	latin1_spanish_ci		No			

Ilustración 40 Tabla solicitud. SIGAME.

Tabla Alertas_Solicitudes. Tabla que recopila datos referentes a las correspondencias entre las solicitudes y las comunidades autónomas, teniendo cada uno un estado. Al insertar una emergencia, es decir, una solicitud, se incluirán en esta tabla 20 tuplas, una por cada comunidad autónoma y otra por la DGPCE, asociadas al identificador de la solicitud que se acaba de insertar. El estado toma el valor "Normal" para todas las comunidades autónomas y "Actualizada" en el caso de la DGPCE. Significa que la solicitud puede ser visitada por todas las comunidades.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	id_solicitud	int(11)			No	0		
☐	comunidad	varchar(25)	latin1_spanish_ci		No			
☐	estado	varchar(15)	latin1_spanish_ci		No			

Ilustración 41 Tabla alertas_solicitudes. SIGAME.

Tabla Estado_Evolución. Tabla que recopila datos referentes al progreso de la emergencia, estará asociada a una solicitud, y contendrá campos más concretos sobre la emergencia como pueden ser número de fallecidos, desaparecidos…, en una fecha dada.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	id	int(11)			No		auto_increment	
☐	id_suceso	int(11)			No	0		
☐	editor	varchar(100)	latin1_swedish_ci		No			
☐	fecha_edicion	varchar(19)	latin1_swedish_ci		No	00/00/0000 00:00:00		
☐	fallecidos	varchar(50)	latin1_swedish_ci		No			
☐	desaparecidos	varchar(50)	latin1_swedish_ci		No			
☐	heridos	varchar(50)	latin1_swedish_ci		No			
☐	evacuados	varchar(50)	latin1_swedish_ci		No			
☐	edif_destruidas	varchar(50)	latin1_swedish_ci		No			
☐	edif_afectadas	varchar(50)	latin1_swedish_ci		No			
☐	cons_esp_afectadas	varchar(50)	latin1_swedish_ci		No			
☐	cortes_carreteras	varchar(50)	latin1_swedish_ci		No			
☐	cortes_ferrocarril	varchar(50)	latin1_swedish_ci		No			
☐	aeropuertos_cerrados	varchar(50)	latin1_swedish_ci		No			
☐	puertos_cerrados	varchar(50)	latin1_swedish_ci		No			
☐	cortes_agua	varchar(50)	latin1_swedish_ci		No			
☐	cortes_electricos	varchar(50)	latin1_swedish_ci		No			
☐	cortes_gas	varchar(50)	latin1_swedish_ci		No			
☐	cortes_telefonicos	varchar(50)	latin1_swedish_ci		No			
☐	mas_info	longtext	latin1_swedish_ci		No			
☐	adjunto	varchar(100)	latin1_swedish_ci		No			

Ilustración 42 Tabla estado_evolucion. SIGAME.

Tabla Recursos_Solicitados. Cada solicitud de emergencia puede necesitar recursos, estos recursos que pueden ser humanos o materiales, los recursos que necesiten en cada momento se irán anotando en esta tabla, los recursos deben existir en la tabla recurso y en el catálogo de recursos. Los campos que almacena son por ejemplo, el nombre del recurso, la fecha en la que se da de alta esa petición de recurso,…

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	id	int(11)			No		auto_increment	
☐	id_solicitud	int(11)			No	0		
☐	id_recurso	int(11)			No	0		
☐	nombre	varchar(30)	latin1_spanish_ci		Sí	*NULL*		
☐	fecha_ini	varchar(19)	latin1_swedish_ci		No	00/00/0000 00:00:00		
☐	fecha_fin	varchar(19)	latin1_swedish_ci		Sí	00/00/0000 00:00:00		
☐	fechaInsercion	varchar(19)	latin1_swedish_ci		No	00/00/0000 00:00:00		
☐	unidades	smallint(6)			No	1		
☐	valor	varchar(30)	latin1_swedish_ci		No			
☐	observaciones	longtext	latin1_spanish_ci		Sí	*NULL*		

Ilustración 43 Tabla recursos_solicitados. SIGAME.

Tabla Oferta. Trata de guardar datos sobre las ofertas que se realizarán para cada solicitud en caso de emergencia, estará asociada a una solicitud. Una oferta en SIGAME responde a un recurso solicitado, es decir, se supone que no se podrá ofertar un recurso que antes no haya sido solicitado, la fecha de la oferta se puede ampliar, y guardará datos de las unidades que ofertan, el estado del recurso, el destinatario, el contacto de la solicitud,… Actualmente, esta tabla almacena la información que en un principio se guardaba en la tabla “recursos_cedidos”, es decir, una oferta se corresponderá por un recurso_cedido si el campo “estado” de la tabla oferta es “cedido”, en caso de que sea “temporal” o “global” significará que ese recurso aun no se ha aceptado como oferta.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	id	int(11)			No		auto_increment	
☐	id_solicitud	int(11)			No	0		
☐	id_recursoSolicitado	int(11)			No	0		
☐	id_recursoPropio	int(11)			No	0		
☐	tipo	text	latin1_spanish_ci		No			
☐	comunidad	varchar(25)	latin1_spanish_ci		No			
☐	udsofertadas	int(11)			No	1		
☐	fecha_ini	varchar(19)	latin1_swedish_ci		No	00/00/0000 00:00		
☐	fecha_fin	varchar(19)	latin1_swedish_ci		No	00/00/0000 00:00		
☐	fecha_registro	varchar(19)	latin1_swedish_ci		No	00/00/0000 00:00		
☐	obs	text	latin1_spanish_ci		No			
☐	condiciones	text	latin1_spanish_ci		No			
☐	estado	varchar(25)	latin1_spanish_ci		No			
☐	fecha_prorroga	varchar(19)	latin1_swedish_ci		No			
☐	destinatario	varchar(50)	latin1_swedish_ci		No			
☐	contacto_solicitud	text	latin1_swedish_ci		Sí	*NULL*		
☐	contacto_oferta	text	latin1_swedish_ci		Sí	*NULL*		

Ilustración 44 Tabla oferta. SIGAME.

También usaremos ciertas tablas pero únicamente para consultar algunos datos, es decir, para verificar que son consistentes.

Tabla Recurso. Esta tabla guarda una descripción concreta de cada recurso, el nombre, las unidades, el titular, si está disponible, ofertado,… en cada momento.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	**id**	int(11)			No		auto_increment	
☐	**tipo**	int(11)			No	0		
☐	**nombre**	varchar(30)	latin1_spanish_ci		Sí	*NULL*		
☐	**unidades**	smallint(6)			No	1		
☐	**valor**	varchar(30)	latin1_swedish_ci		No			
☐	**titular**	varchar(50)	latin1_spanish_ci		No			
☐	**gestor**	varchar(50)	latin1_spanish_ci		No			
☐	**comunidad**	varchar(25)	latin1_spanish_ci		No			
☐	**detalles**	longtext	latin1_spanish_ci		Sí	*NULL*		
☐	**observaciones**	longtext	latin1_spanish_ci		Sí	*NULL*		
☐	**estado**	varchar(50)	latin1_spanish_ci		No			
☐	**disponibles**	smallint(6)			No	0		
☐	**ofertados**	smallint(6)			No	0		
☐	**cedidos**	smallint(6)			No	0		
☐	**destinatario**	varchar(25)	latin1_spanish_ci		No			
☐	**solicitud**	int(11)			No	0		
☐	**oferta**	int(11)			No	0		

Ilustración 45 Tabla recurso. SIGAME.

Tabla Catalogo _ recursos. Este catálogo es común en SIGAME y en ARCE.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	**id**	int(10)		UNSIGNED	No		auto_increment	
☐	**termino**	text	latin1_spanish_ci		No			
☐	**descripcion**	longtext	latin1_spanish_ci		Sí	*NULL*		
☐	**unidad_medida**	varchar(50)	latin1_spanish_ci		Sí	*NULL*		

Ilustración 46 Tabla catalogo_recursos. SIGAME.

La siguiente imagen muestra el modelo relacional de la base de datos SIGAME, para las tablas que utilizaremos en el proyecto, sólo se muestran las claves primarias con el fin de que sea más clara.

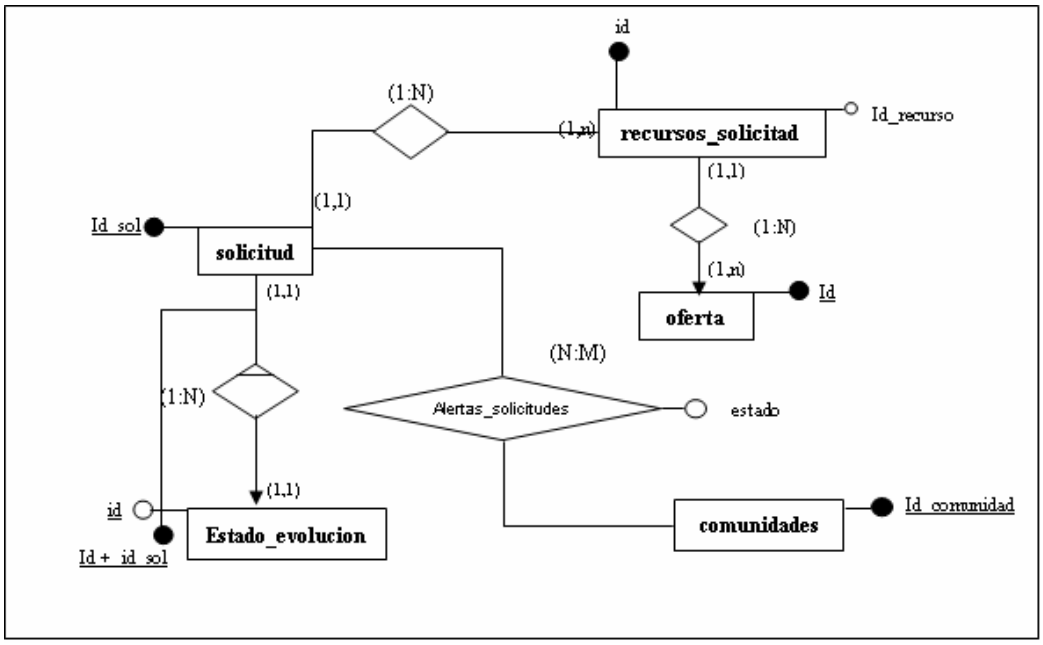


Ilustración 47 Modelo entidad-relación SIGAME

En ARCE al dar de alta una emergencia, o insertar aportaciones… se rellenan una serie de formularios que recogen los datos y los almacenan en unas tablas que es denominan de "primera_solicitud" son una primera aproximación de los datos de la emergencia, son para hacerse a la idea de lo que está ocurriendo hasta las primeras 48 horas, una vez pasada esta aproximación los datos son trasladados a las tablas que usaremos en este proyecto, el motivo de no usar las tablas de "primera_solicitud", es que al ser emergencias que provienen del sistema de emergencia SIGAME, estos datos ya son reales.

La definición de las tablas de la base de datos ARCE se muestra a continuación:

Table	Owner	Tablespace	Estimated row count	Actions						Comment
aeropuerto	postgres		319	Browse	Select	Insert	Empty	Drop	Vacuum	
ambito	postgres		48	Browse	Select	Insert	Empty	Drop	Vacuum	
aportacion	postgres		10	Browse	Select	Insert	Empty	Drop	Vacuum	
aportacion_cursada	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
aportacion_primera_solicitud	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
aportacion_ps_cursada	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
area	postgres		1	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_ambito	postgres		48	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_elemento_ps	postgres		32	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_emergencia	postgres		11	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_evento	postgres		11	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_lugar	postgres		505	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_primera_solicitud	postgres		1	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_recurso	postgres		4020	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_recurso_emergencia	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
cod_termino	postgres		45008	Browse	Select	Insert	Empty	Drop	Vacuum	
comunicacion	postgres		162	Browse	Select	Insert	Empty	Drop	Vacuum	
comunicado	postgres		16	Browse	Select	Insert	Empty	Drop	Vacuum	
conversacion	postgres		1	Browse	Select	Insert	Empty	Drop	Vacuum	
conversacion_historico	postgres		10	Browse	Select	Insert	Empty	Drop	Vacuum	
coordinador	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
debug_inserta	postgres		373567	Browse	Select	Insert	Empty	Drop	Vacuum	
discusion_glosario	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
elemento_de_una_ps	postgres		10	Browse	Select	Insert	Empty	Drop	Vacuum	
elemento_ps	postgres		32	Browse	Select	Insert	Empty	Drop	Vacuum	
emergencia	postgres		11	Browse	Select	Insert	Empty	Drop	Vacuum	
emergencias_entidad	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
entidad	postgres		28	Browse	Select	Insert	Empty	Drop	Vacuum	
estacion	postgres		5	Browse	Select	Insert	Empty	Drop	Vacuum	
evento	postgres		22	Browse	Select	Insert	Empty	Drop	Vacuum	
glosario	postgres		50949	Browse	Select	Insert	Empty	Drop	Vacuum	
idioma	postgres		2	Browse	Select	Insert	Empty	Drop	Vacuum	
lugar_entrega	postgres		4	Browse	Select	Insert	Empty	Drop	Vacuum	
lugar_entrega_ps	postgres		2	Browse	Select	Insert	Empty	Drop	Vacuum	
mensajes	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
mensajes_historico	postgres		252	Browse	Select	Insert	Empty	Drop	Vacuum	
noticia	postgres		1519	Browse	Select	Insert	Empty	Drop	Vacuum	
pais	postgres		22	Browse	Select	Insert	Empty	Drop	Vacuum	
paises_comunicacion	postgres		2612	Browse	Select	Insert	Empty	Drop	Vacuum	
paises_comunicado	postgres		308	Browse	Select	Insert	Empty	Drop	Vacuum	
paises_conversacion	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
paises_historico	postgres		27	Browse	Select	Insert	Empty	Drop	Vacuum	
primera_solicitud	postgres		1	Browse	Select	Insert	Empty	Drop	Vacuum	
propuesta_glosario	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
puerto	postgres		182	Browse	Select	Insert	Empty	Drop	Vacuum	
recurso	postgres		3565	Browse	Select	Insert	Empty	Drop	Vacuum	
recursos_aportados	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
recursos_aportados_ps	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
recursos_solicitados	postgres		7	Browse	Select	Insert	Empty	Drop	Vacuum	
relaciones_recursos	postgres		3566	Browse	Select	Insert	Empty	Drop	Vacuum	
roles	postgres		255	Browse	Select	Insert	Empty	Drop	Vacuum	
roles_comunicacion	postgres		1143	Browse	Select	Insert	Empty	Drop	Vacuum	
roles_comunicado	postgres		39	Browse	Select	Insert	Empty	Drop	Vacuum	
roles_conversacion	postgres		0	Browse	Select	Insert	Empty	Drop	Vacuum	
roles_historico	postgres		12	Browse	Select	Insert	Empty	Drop	Vacuum	
solicitud	postgres		3	Browse	Select	Insert	Empty	Drop	Vacuum	
termino	postgres		45008	Browse	Select	Insert	Empty	Drop	Vacuum	
usuario	postgres		213	Browse	Select	Insert	Empty	Drop	Vacuum	

Ilustración 48 Tablas de la Base de datos ARCE.

Las tablas que usaremos en el proyecto serán las citadas a continuación, como en el caso anterior existen tablas que sólo utilizaremos a modo de consulta:

Tabla Emergencia. La tabla Emergencia almacena los datos de una emergencia, datos concretos, puesto que ya no es "primera_solicitud", en esta tabla se puede ver la evolución que va tomando una emergencia, puesto que posee un campo traza, el valor más alto de este campo es el que marca la última modificación, se guardarán datos como los daños causados por cada suceso.

Column	Type	Not Null	Default	Actions	Comment
codigo_emergencia	integer	NOT NULL		Alter Drop	
traza	integer	NOT NULL	1	Alter Drop	
estado	character(1)	NOT NULL	'A'::bpchar	Alter Drop	
usuario	character varying(20)	NOT NULL		Alter Drop	
fecha	abstime	NOT NULL		Alter Drop	
t_natural	character(1)	NOT NULL	'S'::bpchar	Alter Drop	
t_tecno	character(1)	NOT NULL	'N'::bpchar	Alter Drop	
t_otro	character(1)	NOT NULL	'N'::bpchar	Alter Drop	
t_ejercicio	character(1)	NOT NULL	'N'::bpchar	Alter Drop	
causa	character varying(20)	NOT NULL		Alter Drop	
poblaciones	text	NOT NULL		Alter Drop	
afectados	integer	NOT NULL		Alter Drop	
descripcion	text	NOT NULL		Alter Drop	
fecha_estimacion	abstime			Alter Drop	
muertas	integer			Alter Drop	
heridas	integer			Alter Drop	
desaparecidas	integer			Alter Drop	
atrapadas	integer			Alter Drop	
aisladas	integer			Alter Drop	
evacuadas	integer			Alter Drop	
sin_techo	integer			Alter Drop	
albergar	integer			Alter Drop	
alimentar	integer			Alter Drop	
danos_viviendas	text			Alter Drop	
danos_edi_ofi	text			Alter Drop	
danos_hospitales	text			Alter Drop	
danos_agua	text			Alter Drop	
danos_electricidad	text			Alter Drop	
danos_gas	text			Alter Drop	
danos_telefonia	text			Alter Drop	
danos_carreteras	text			Alter Drop	
situ_alcantarillado	text			Alter Drop	
situ_alimentos	text			Alter Drop	
situ_sanitaria	text			Alter Drop	
ayuda_exterior	character(1)	NOT NULL	'S'::bpchar	Alter Drop	
informacion	text			Alter Drop	
fichero	character varying(100)			Alter Drop	
restricciones	text			Alter Drop	
dv	integer			Alter Drop	
de	integer			Alter Drop	
dh	integer			Alter Drop	
sa	integer			Alter Drop	
ss	integer			Alter Drop	
ds	integer			Alter Drop	
sr	integer			Alter Drop	
di	integer			Alter Drop	
dg	integer			Alter Drop	
dt	integer			Alter Drop	
dr	integer			Alter Drop	
fecha_emision	abstime			Alter Drop	

Ilustración 49 Tabla emergencia. ARCE.

Tabla Solicitud. Esta tabla está asociada a la tabla Emergencia, se centra en el alta de una solicitud de una emergencia concreta, en una fecha, por un usuario… para que quede constancia.

Column	Type	Not Null	Default	Actions	Comment
codigo_solicitud	integer	NOT NULL	nextval(('sec_solicitud'::text)::regclass)	Alter Drop	
fecha	abstime	NOT NULL		Alter Drop	
codigo_emergencia	integer	NOT NULL		Alter Drop	
usuario	character varying(20)	NOT NULL		Alter Drop	
pais	character varying(20)	NOT NULL		Alter Drop	
estado	character(1)	NOT NULL	'P'::bpchar	Alter Drop	

Ilustración 50 Tabla solicitud. ARCE.

Tabla Recursos_Solicitados. Es una tabla que guarda los datos asociados a una solicitud para un recurso concreto y una cantidad.

Column	Type	Not Null	Default	Actions	Comment
codigo_solicitud	integer	NOT NULL		Alter Drop	
codigo_recurso	integer	NOT NULL		Alter Drop	
cantidad	integer	NOT NULL		Alter Drop	

Ilustración 51 Tabla recursos_solicitados. ARCE.

Tabla Aportación. Trata de almacenar datos sobre una aportación, que se realiza a una solicitud concreta, sabiendo la fecha, el estado del emisor,…

Column	Type	Not Null	Default	Actions	Comment
codigo_aportacion	integer	NOT NULL	nextval(('sec_aportacion'::text)::regclass)	Alter Drop	
aportacion_cursada	integer	NOT NULL		Alter Drop	
fecha	abstime	NOT NULL		Alter Drop	
estado_receptor	character(1)	NOT NULL	'P'::bpchar	Alter Drop	
estado_emisor	character(1)	NOT NULL	'A'::bpchar	Alter Drop	
codigo_solicitud	integer	NOT NULL		Alter Drop	
usuario	character varying(20)	NOT NULL		Alter Drop	
lugar	integer	NOT NULL		Alter Drop	
comentarios	text			Alter Drop	

Ilustración 52 Tabla aportacion. ARCE.

Tabla Aportacion_cursada. Esta tabla se encarga de almacenar los datos de las aportaciones cursadas, datos como las fechas de inicio y fin de la aportación, el país,…Insertaremos en esta tabla cuando introduzcamos una tupla en la tabla aportación.

Column	Type	Not Null	Default	Actions	Comment
codigo_aportacion	integer	NOT NULL	nextval(('sec_aportacion_cursada'::text)::regclass)	Alter Drop	
fecha_inicio	abstime	NOT NULL		Alter Drop	
fecha_fin	abstime			Alter Drop	
pais	integer	NOT NULL		Alter Drop	
entidad	integer	NOT NULL		Alter Drop	

Ilustración 53 Tabla aaportación_cursada. ARCE.

Tabla Recursos_Aportados. Al igual que los recursos_solicitados, solo guarda el identificador del recurso, el código de la solicitud a la que pertenece y una cantidad. Marca los recursos que han sido aportados ya.

Column	Type	Not Null	Default	Actions	Comment
codigo_aportacion	integer	NOT NULL		Alter Drop	
codigo_recurso	integer	NOT NULL		Alter Drop	
aportacion	integer	NOT NULL		Alter Drop	

Ilustración 54 Tabla recursos_aportados. ARCE.

Tabla Lugar_Entrega. Es la que almacena para cada solicitud, el lugar donde se producirá la entrega de los recursos aportados.

Column	Type	Not Null	Default	Actions	Comment
codigo_lugar_entrega	integer	NOT NULL	nextval(('sec_lugar_entrega'::text)::regclass)	Alter Drop	
codigo_solicitud	integer	NOT NULL		Alter Drop	
codigo_lugar	integer	NOT NULL		Alter Drop	

Ilustración 55 Tabla lugar_entrega. ARCE.

Tabla Cod_Lugar. En ARCE, es importante saber el lugar de entrega en cada momento, sin embargo, en SIGAME no se le da tanta importancia, esto será estudiado más adelante. Esta tabla y las siguientes muestran una jerarquía, es decir, existirá para cada solicitud uno o varios lugares de entrega pero siempre será un aeropuerto, o un área, o una estación, o un puerto.

Column	Type	Not Null	Default	Actions	Comment
codigo_lugar	integer	NOT NULL	nextval(('sec_lugar'::text)::regclass)	Alter Drop	
borrado	character(1)	NOT NULL	'N'::bpchar	Alter Drop	

Ilustración 56 Tabla cod_lugar. ARCE.

Tabla Aeropuerto. Guarda los aeropuertos de los países de la base de datos.

Column	Type	Not Null	Default	Actions		Comment
codigo_lugar	integer	NOT NULL		Alter	Drop	
aeropuerto	character varying(50)			Alter	Drop	
municipio	character varying(50)			Alter	Drop	
muneces	character varying(7)			Alter	Drop	
nombre	character varying(50)	NOT NULL		Alter	Drop	
localizacion	character varying(60)			Alter	Drop	
tp	integer			Alter	Drop	
tc	integer			Alter	Drop	
pista1	character varying(7)			Alter	Drop	
pista2	character varying(7)			Alter	Drop	
lpista1	integer			Alter	Drop	
apista1	integer			Alter	Drop	
lpista2	integer			Alter	Drop	
apista2	integer			Alter	Drop	
direccion	character varying(50)			Alter	Drop	
c_postal	character varying(5)			Alter	Drop	
localidad	character varying(50)			Alter	Drop	
pr	character varying(2)			Alter	Drop	
pref	character varying(3)			Alter	Drop	
tel1	character varying(7)			Alter	Drop	
tel2	character varying(7)			Alter	Drop	
horario	character varying(50)			Alter	Drop	
horario2	character varying(50)			Alter	Drop	
arp_lat	character varying(7)			Alter	Drop	
arp_long	character varying(7)			Alter	Drop	
arp_z	integer			Alter	Drop	
propiedad	character varying(50)			Alter	Drop	
f_actual	character varying(10)			Alter	Drop	
pais	integer	NOT NULL		Alter	Drop	
icao	character(4)			Alter	Drop	

Ilustración 57 Tabla aeropuerto. ARCE.

Tabla Area. Guarda las áreas de los países de la base de datos.

Column	Type	Not Null	Default	Actions	Comment
codigo_lugar	integer	NOT NULL		Alter Drop	
nombre	character varying(40)	NOT NULL		Alter Drop	
descripcion	text			Alter Drop	
pais	integer	NOT NULL		Alter Drop	

Ilustración 58 Tabla area. ARCE.

Tabla Estacion. Guarda estaciones de tren de los países de la base de datos.

Column	Type	Not Null	Default	Actions	Comment
codigo_lugar	integer	NOT NULL		Alter Drop	
nombre	character varying(40)	NOT NULL		Alter Drop	
numero_andenes	integer			Alter Drop	
ancho_via1	integer			Alter Drop	
ancho_via2	integer			Alter Drop	
pais	integer	NOT NULL		Alter Drop	

Ilustración 59 Tabla estacion. ARCE.

Tabla Puerto. Guarda los puertos de los países de la base de datos.

Column	Type	Not Null	Default	Actions		Comment
codigo_lugar	integer	NOT NULL		Alter	Drop	
nombre	character varying(40)	NOT NULL		Alter	Drop	
oceano	character varying(20)	NOT NULL		Alter	Drop	
provincia	character varying(40)			Alter	Drop	
unctad	character varying(6)	NOT NULL		Alter	Drop	
latitud	character varying(8)	NOT NULL		Alter	Drop	
longitud	character varying(8)	NOT NULL		Alter	Drop	
pais	integer	NOT NULL		Alter	Drop	

Ilustración 60 Tabla puerto. ARCE.

Tabla Roles. Almacena el login de los usuarios y su rol correspondiente. Esta tabla la usaremos como consulta, para comprobar los roles de los usuarios puesto que sólo los usuarios de nivel 4 serán los que den de alta las emergencias que se migrarán a Arce.

Column	Type	Not Null	Default	Actions		Comment
login	character varying(20)	NOT NULL		Alter	Drop	
rol	integer	NOT NULL		Alter	Drop	

Ilustración 61 Tabla roles. ARCE.

Tabla Usuario. Recoge todos los datos referentes a los usuarios, esta tabla también se usará para realizar las comprobaciones oportunas a la hora de migrar la información.

Column	Type	Not Null	Default	Actions	Comment
login	character varying(20)	NOT NULL		Alter Drop	
password	character varying(20)	NOT NULL		Alter Drop	
denom_cargo	character varying(50)	NOT NULL		Alter Drop	
denom_responsable	character varying(50)	NOT NULL		Alter Drop	
area_operativa	character(1)	NOT NULL		Alter Drop	
area_formativa	character(1)	NOT NULL		Alter Drop	
area_economica	character(1)	NOT NULL		Alter Drop	
area_tec_riesgos	character(1)	NOT NULL		Alter Drop	
riesgo_natural	character(1)	NOT NULL		Alter Drop	
riesgo_tecnologico	character(1)	NOT NULL		Alter Drop	
riesgo_antropico	character(1)	NOT NULL		Alter Drop	
especialidad	character varying(50)	NOT NULL		Alter Drop	
nombre	character varying(30)	NOT NULL		Alter Drop	
apellidos	character varying(30)	NOT NULL		Alter Drop	
direccion	character varying(100)	NOT NULL		Alter Drop	
codigo_postal	character varying(20)	NOT NULL		Alter Drop	
localidad	character varying(50)	NOT NULL		Alter Drop	
pep	character varying(100)	NOT NULL		Alter Drop	
telefono	character varying(20)	NOT NULL		Alter Drop	
telefono_directo	character varying(20)	NOT NULL		Alter Drop	
telefono_alt	character varying(20)	NOT NULL		Alter Drop	
telefono_celular	character varying(20)	NOT NULL		Alter Drop	
fax	character varying(20)	NOT NULL		Alter Drop	
fax_directo	character varying(20)	NOT NULL		Alter Drop	
fax_alt	character varying(20)	NOT NULL		Alter Drop	
correo	character varying(50)	NOT NULL		Alter Drop	
entidad	integer	NOT NULL		Alter Drop	
borrado	character(1)	NOT NULL	'N'::bpchar	Alter Drop	

Ilustración 62 Tabla usuario. ARCE.

Tabla Entidad. Recoge todos los datos referentes a las entidades, esta tabla también se usará para realizar las comprobaciones oportunas a la hora de migrar la información.

Column	Type	Not Null	Default	Actions	Comment
codigo_entidad	integer	NOT NULL	nextval(('sec_entidad'::text)::regclass)	Alter Drop	
nombre	character varying(120)	NOT NULL		Alter Drop	
siglas	character varying(20)	NOT NULL		Alter Drop	
dependencia	character varying(50)	NOT NULL		Alter Drop	
descripcion	text	NOT NULL		Alter Drop	
direccion	character varying(200)	NOT NULL		Alter Drop	
codigo_postal	character varying(20)	NOT NULL		Alter Drop	
localidad	character varying(50)	NOT NULL		Alter Drop	
pep	character varying(100)	NOT NULL		Alter Drop	
telefono	character varying(20)	NOT NULL		Alter Drop	
fax	character varying(20)	NOT NULL		Alter Drop	
url	character varying(100)	NOT NULL		Alter Drop	
correo	character varying(50)	NOT NULL		Alter Drop	
tipo_entidad	integer	NOT NULL		Alter Drop	
idioma	character varying(3)	NOT NULL		Alter Drop	
pais	integer	NOT NULL		Alter Drop	
presidencia	integer			Alter Drop	
autorizada	character(1)	NOT NULL	'S'::bpchar	Alter Drop	
borrado	character(1)	NOT NULL	'N'::bpchar	Alter Drop	

Ilustración 63 Tabla entidad. ARCE.

Si vemos la estructura referente a las emergencias la Base de Datos del Sistema de Emergencia ARCE, vemos que es la siguiente (la siguiente imagen refleja únicamente las tablas de la Base de Datos que vamos a utilizar en el proyecto, y la clave primaria de cada tabla, con el motivo de ver lo más esencial):

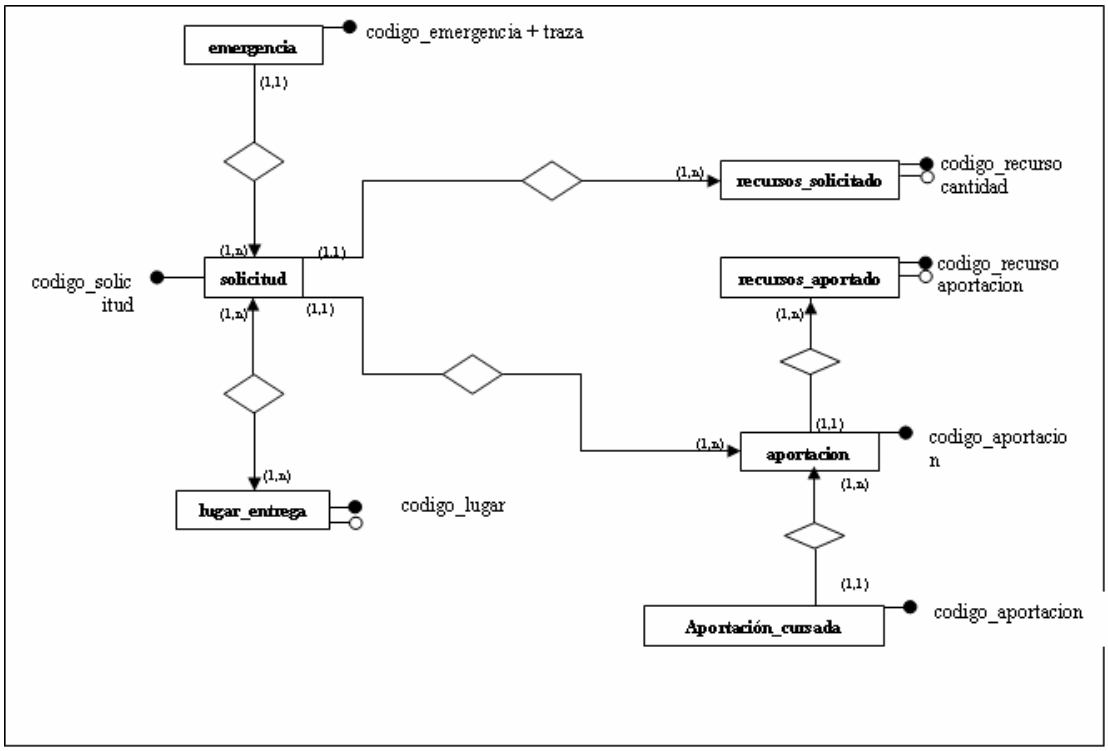


Ilustración 64 Modelo entidad-relación ARCE

A continuación mostraremos la jerarquía de los lugares (no se incluirán las claves primarias):

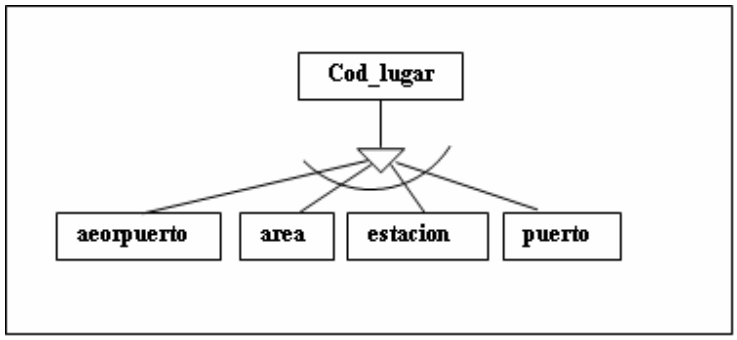


Ilustración 65 Jerarquía lugares